Reconocimiento de edad con corrección de pose basado en técnicas de aprendizaje de máquina

Proyecto de grado presentado como requisito parcial para aspirar al título de Ingeniero Electrónico

Estudiantes:
David Stiven Fresneda Hernández
César Augusto Orrego Hurtado

Supervisor: M. Sc. Julián Gil González

Universidad Tecnológica de Pereira
Facultad de Ingenierías
Facultad de Ingeniería Eléctrica, Electrónica, Física, Sistemas y Computación
Programa de Ingeniería Electrónica
Pereira, Risaralda
2020

Agradecimientos

Gracias a nuestros padres, que fueron los mayores promotores durante este proceso, por la dedicación, paciencia y confianza que depositaron en nosotros pese a todos los inconvenientes presentados. Por la ayuda incondicional y esfuerzos realizados durante todo el proceso de formación sin contar con cada uno de los sacrificios tomados para poder llegar hasta este punto. Gracias a Dios, que fue el principal apoyo y motivador para cada día continuar con el proceso y ayudarnos a ver la luz frente a tan largo camino.

Especial agradecimiento al M. Sc. Julián Gil González por haber tenido la confianza en nosotros llevando a cabo este proyecto de grado, además de su apoyo constante y preciso.

Gracias a la Universidad Tecnológica de Pereira, por habernos permitido formarnos y en ella desarrollarnos como personas y profesionales, enseñándonos siempre los valores que priman en cada uno de los procesos de aprendizaje; gracias a todas las personas que fueron partícipes de este proceso, ya sea de manera directa o indirecta, por el apoyo, enseñanzas y experiencias vividas tanto dentro como fuera de la institución.

Resumen

El presente trabajo tiene como fin realizar una metodología que permite llevar a cabo un reconocimiento de edad con corrección de pose basado en técnicas de aprendizaje de máquina a partir de imágenes. Estas imágenes son introducidas al sistema el cual determina un intervalo de edad para la persona en cuestión; esto con el fin de brindar una herramienta tecnológica que permita recolectar información como los gustos, edad, tendencias, ideologías, entre otras características, permitiendo su implementación e incursión en diversas aplicaciones como transporte, medicina, control de accesos, entretenimiento para la selección del contenido, el marketing; entre otras áreas.

Esta metodología se lleva a cabo de forma inicial con el preprocesamiento de la imagen, es decir se realiza un acondicionamiento apropiado para poder tener una mayor efectividad a la hora de extraer la información relevante; para ello en la imagen se realiza un reconocimiento del rostro, se determina el ángulo de pose con respecto al eje horizontal y se corrige. Se determinan varias regiones de interés hallando de tal forma características globales y locales que permiten discriminar con mayor grado la edad a través de procesos matemáticos de descripción como Modelos de Apariencia Actica (AAM), Gabor, Transformada Wavelet (TW) y Patrones Binarios Locales (LBP).

Posterior a esta recolección de datos se realiza una simplificación de la información a través de métodos estadísticos como el Análisis de Componentes Principales (PCA), esta etapa permite elegir entre las diferentes características las que aportan mayor cantidad de informacíon; es decir, povee una forma para la reducción de dimensionalidad del conjunto dado el menor número de componentes posible y finalmente para la clasificación se subdividió el conjunto de imágenes en cuatro rangos de edad de acuerdo con la distribución de datos, con el objetivo de no sobrecargar ninguna de las clases y poder así tener suficientes datos para el entrenamiento y prueba, esto también permite tener un grupo para niños, adultos jóvenes, adultos y personas mayores, especialmente útil para el modelamiento del envejecimiento, con esto se obtiene una clasificación de la información; a través de los clasificadores como regresión logística (LR), Máquinas de Soporte Vectorial (SVM) y Procesos Gaussianos (GP) para determinar cual es el más efectivo y calcular el intervalo de edad además de evaluar el desempeúo de cada uno ellos.

Con esta metodología implementada se logra evidenciar aspectos relevantes para tener un buen rendimiento en métodos de aprendizaje de máquina específicamente con la correcccón de la pose frente a la cámara, además de la normalización de las imágenes para tener aproximadamente el mismo tiempo de procesamiento y dimesionalidad entre cada imágen.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Preliminares

1.1. Introducción

En la actualidad la tecnología está inmersa en muchas de las actividades que realiza el ser humano en su cotidianidad, de modo que hay una gran cantidad de sistemas que interactúan con los usuarios. Estos sistemas recolectan información como los gustos, la edad, tendencias, ideologías y sitios que frecuentan, entre otros [1]. En otras palabras es de vital importancia que los sistemas puedan interpretar la información de su entorno para una mejor interacción con el usuario.

Uno de los tipos de información que recolectan los sistemas descritos anteriormente es la edad de los usuarios a partir de imágenes, lo cual tiene muchas aplicaciones e incursión en campos de seguridad y protección, transporte, medicina, control de accesos, entretenimiento para la selección del contenido y el marketing [2]. Es así como el rostro representa una gran fuente de información de identidad, emociones, y la edad. Logrando obtener del rostro la textura de la piel, arrugas, machas y cicatrices, características importantes para la identificación de la edad, así como la forma de la cara que permite inferir el crecimiento craneofacial de las personas con el fin de determinar su edad [3].

El reconocimiento facial ha sido un tema muy investigado y con un gran número de soluciones [4]. Sin embargo, el proceso para determinar la edad a partir de imágenes faciales ha sido menos abordado y es aún una problemática desafiante debido a diversos factores como: *i*) la iluminación y ubicación de los rostros ante la cámara [5] [6]. *ii*) el hecho de que la edad es una variable que depende de diversos factores como la salud, estilo de vida, el entorno e incluso el género. De esta manera dos personas con edades similares pueden tener procesos de envejecimiento distintos.

De acuerdo a lo anterior, estos problemas se pueden minimizar a través de sistemas automáticos basados en técnicas de Aprendizaje de Maquina Supervisado. Para este caso, se ve como un problema de clasificación de múltiples clases [7] en la cual permitirá determinar en qué clase etario o grupo de edad se encuentra la persona. El problema del reconocimiento automático de la edad a partir del análisis de una imagen de la persona desde la vista de un problema de reconocimiento de visión por computador, se logra abordar en tres pasos fundamentales que son: *i*) la obtención y el preprocesamiento de la imagen, *ii*) la extracción de características y *iii*) la clasificación en rangos de edad.

De tal forma que lo anterior plantea la metodología y los pasos a seguir para cumplir el objetivo del proyecto. En primer lugar en la etapa de preprocesamiento se basa en obtener el área del rostro de la persona en la imagen, eliminando distorsiones y omitir zonas que no aportan información para la clasificación [7]. Después se busca realizar la alineación del rostro para analizar imágenes en condiciones más desafiantes al tratar de corregir la pose de la persona permitiendo obtener características antes ocultas para así mejorar el rendimiento del clasificador [8].

En segundo lugar, una vez se tiene solo la zona del rostro, se procede a extraer las características más relevantes de la cara y que proporcionen mayor información para determinar la edad. A pesar de que el proceso de envejecimiento es diferente para cada persona, si hay patrones generales en común que lo describen. Hay dos diferentes etapas durante el desarrollo facial, desde el nacimiento hasta la edad adulta, el mayor cambio es el crecimiento craneofacial, mientras que en la edad adulta hasta la vejez, el cambio más perceptible es el envejecimiento de la piel [4].

Es así que se define como descriptores los que permitan modelar estas dos características, los modelos de apariencia activa (AAM) permiten obtener información de la forma y apariencia del cráneo, y para las características de la piel se definen los descriptores de textura como los Patrones Binarios Locales (LBP), la Transformada Wavelet, y los filtros Gabor Wavelet [9]. Una vez transformada cada una de las imágenes en un conjunto de datos (matriz de características), se realiza la selección de características más relevantes para obtener un mejor rendimiento y un menor costo computacional al reducir las dimensiones de la matriz por medio de la técnica de Análsis de Componentes Principales (PCA).

De esta forma se ha convertido la imagen en una representación de modelos matemáticos y estadísticos. La etapa final para la estimación de la edad se toma desde un problema de clasificación multiclase, de esta forma se definen varios grupos etarios. En esta etapa se implementan, entrenan y validan los métodos de regresión logística, máquina de soporte vectorial y por ultimo procesos Gaussianos para la clasificación de 4 grupos de edad.

Por lo tanto, en este trabajo se desarrolla un algoritmo para el reconocimiento de grupos de edad con corrección de pose del rostro de una persona en la imagen. Implementando las características globales y locales para modelar el proceso de envejecimiento y logrando una mejor eficiencia con la reducción de características. El rendimiento de cada uno de los clasificadores propuestos es evaluado con la base de datos FG-NET y compensando las clases de adultos y adultos mayores con imágenes de MORPH-II. La combinación de las dos bases de datos crea una distribución apropiada para cada clase, permitiendo una mejora en el rendimiento. Consiguiendo que los resultados experimentales sean similares a los diferentes enfoques presentados en el estado del arte.

1.2. Objetivos

1.2.1. Objetivo General

Diseñar e implementar una metodología para el reconocimiento de edad en imágenes de rostros basado en técnicas de aprendizaje de máquina que sea robusto a variaciones en la pose de los rostros.

1.2.2. Objetivos Específicos

1. Implementar una metodología de preprocesamiento con el fin de corregir la distorsión presente en la pose de los rostros recolectados.

2. Implementar una metodología para la extracción y selección de características que permita discriminar entre los diferentes rangos de edades tenidos en cuenta en este trabajo.

3. Implementar un esquema de clasificaciíon basado en regresión logística (LR), máquina de soporte vectorial (SVM) y en procesos Gaussianos (GP).

1.3. Estado del Arte

El rostro humano proporciona una gran cantidad de información sobre una persona. En este sentido hay unas características favorables que hacen viable la implementación de sistemas que realicen análisis facial, al igual que surgen ciertos desafíos. Algunas ventajas o características es la simetría del rostro, lo que permite beneficiar las tareas de localización o detección facial, la cara tiene un alto poder discriminante al tener una gran cantidad de rasgos y alta variabilidad lo que beneficia las tareas de clasificación a partir de características [10]. Entre los inconvenientes o desafíos que se presentan para estas técnicas es la variabilidad de la cara debido a las expresiones faciales o gestos, además de que se trabajan en ambientes no controlados lo que puede llevar a cambios en la pose, oclusión (gafas, gorras), calidad de la imagen, iluminación, hacen que el sistema pierda fiabilidad [11].

En los últimos años se ha realizado un mayor número de investigaciones y diferentes enfoques en la clasificación y estimación de la edad a partir del procesamiento de imágenes faciales [4]. Tratando de implementar nuevas técnicas y métodos para mejorar el rendimiento frente a los diversos factores desafiantes como la iluminación, la pose del rostro frente a la cámara, el entorno y el error de clasificación en la cercanía de las fronteras de los grupos clasificados.

A grandes rasgos el procedimiento o estructura de la determinación de la edad puede dividirse en 3 etapas generales, primero la etapa del preprocesamiento el cual se desencadena con la adquisición de la imagen para luego prepararla haciendo la detección facial, el recorte, escalado, y alineación del rostro. En la segunda etapa se realiza la extracción y selección de características más relevantes del rostro y en la última etapa se clasifica en rangos etarios a partir de las características extraídas.

1.3.1. Preprocesamiento

En esta primer etapa, inicialmente consiste en la detección del rostro en la imagen, es decir hallar áreas o zonas de la imagen que correspondan a un rostro de una persona, para obtener solo esta zona y aislarla del resto que no son deseadas, como el fondo. Es de gran importancia realizar una buena detección para que las demás fases tengan un buen rendimiento, ya que una mala detección conllevaría a un error en las siguientes etapas [10].

En el preprocesamiento cada una de las imágenes se ajustan para poder tener un sistema estandarizado, eliminando secciones que no son de interés debido a que el área principal es el rostro y no otra parte del cuerpo, o eliminando hasta el mismo entorno donde se encuentre dicha persona. En el estado del arte han sido implementados varias técnicas, para esto se define primero un punto de referencia y a partir de este se realiza el reconocimiento del rostro y el recorte, en [7] primero se identifican los ojos en la imagen por medio de *Haar*

feature-based cascade classifier, después se identifica el ovalo facial, se recorta y rota (si es necesario) para que quede alineado con los ojos.

Por otra parte, en [9] toman como referencia la distancia media que hay entre los ojos, de forma que se puede trazar una serie de puntos que delimitan las facciones de la cara y posteriormente seccionar las áreas de interés las cuales se les asigna cierto valor que definen la ubicación dentro de la imagen. Además, es necesario para obtener buenos resultados en el rendimiento del clasificador, eliminar el ruido en las imágenes de esta manera se añaden, reducen o eliminan componentes de iluminación, color, tamaño de la imagen, entre otras características [11], y en [7] aplican el filtros Bilateral y el filtro Gabor.

Una de las técnicas del estado del arte más populares y eficientes al tener bajo costo computacional a la hora de detectar objetos y más especialmente para detectar rostros, es el algoritmo de Viola-Jones [12]. Este algoritmo se basa en una serie de descriptores débiles denominados Haar-like features [13], son características muy simples que se busca en la imagen y que consisten en la diferencia de intensidades luminosas entre las regiones rectangulares adyacentes. Cada uno de estos clasificadores se agrupan en cascada empleando un algoritmo adaptativo de machine learning denominado AdaBoost "*adaptative boosting*" [14], de esta forma se consigue un alto rendimiento en la detección final. Adaboost es un método de clasificación que es particularmente adecuado cuando el número de características del descriptor es muy elevado como va a suceder con las características de HAAR. Las características de HAAR se obtienen después de aplicar un conjunto de filtros por toda la imagen a diferentes escalas. De esta manera se tiene un número muy elevado de características del orden de más de 100.000 por imagen. El clasificador va a tener que tratar con este número tan elevado de características y para ello se utiliza Adaboost, que permite mientras se lleva a cabo el aprendizaje también determinar y seleccionar las características que son más relevantes para la clasificación. El resultado de aplicar cada filtro HAAR en cada posición y escala va a ser una componente del vector final de características.

Una vez se obtiene el rostro con alguna de las técnicas expuestas anteriormente, se continúa con el procedimiento para preparar la imagen para obtener un buen rendimiento, por lo tanto se realiza el escalado, la rotación, el recorte y la ecualización del histograma.

- En el escalado se busca que todas las imágenes a procesar por el sistema, tengan las mimas dimensiones o cantidad de píxeles, por lo tanto la ventana de detección del rostro se aumenta o reduce. Para esto se utiliza la distancia entre los centros de los ojos para obtener un ratio para escalar uniformemente la imagen [10] [15].
- La alineación se realiza por medio de las coordenadas de los ojos. Se obtiene el ángulo de giro que tiene el rostro en la imagen respecto al eje x. Al rotar la imagen se compensa este ángulo hasta que sea cero y el rostro quede alineado con el eje x. Al tener rostros sin giro, se obtienen resultados positivos debido a que el algoritmo desarrollado no tendrá

pérdida en la sección de reconocimiento ya que la orientación es la misma para cada una de las imágenes, este procedimiento se lleva a cabo a cada una de las imágenes de la base de datos de manera uniforme [6] [8].

- Recorte: Una vez se tenga alineado y escalado el rostro, se realiza el recorte de la imagen. Esto permite disminuir el tiempo de procesamiento que requiere cada imagen además de tomar la(s) región(es) de interés con las cuales se llevará a cabo todo el proceso de descripción y clasificación de edad [10] [15].
- Ecualización del histograma: Las imágenes pueden presentar variabilidad en la luminosidad y en el contraste lo que produce que imágenes similares sean muy diferentes respecto al valor de intensidad de sus píxeles. Mediante la ecualización de su histograma, se pretende que las imágenes que tienen la mayor parte de sus valores de intensidad concentrados en una zona reducida del histograma, pasen a extenderse por todo el rango de valores del mismo. Esto resulta en imágenes con mayor contraste y con menor variabilidad lumínica entre ellas [10].

1.3.2. Extracción de Características

La extracción de características se emplea para obtener la información de la cara que resulta relevante para la estimación de la edad, en las últimas décadas se han desarrollado un gran número de técnicas y algoritmos para este fin. En esta área se resaltan principalmente dos enfoques, en primer lugar están las técnicas basadas en apariencia, que consisten en tratar el problema en un análisis de espacio en donde se aplican diferentes técnicas estadísticas. En segundo lugar están las técnicas basadas en modelos, que extraen las características tanto de la forma del rostro como de la textura [10]. En los trabajos relacionados a la estimación y clasificación de la edad se utilizan los enfoques basados en modelos.

En relación a lo anterior, por medio de las técnicas basadas en modelos, en esta etapa se parte de la imagen normalizada del preprocesamiento y se obtendrá como salida un vector de características. Un modelo matemático con la información codificada de las características relevantes de la edad en el rostro, como lo son las arrugas, la textura de la piel, manchas y la forma del rostro, así como también las zonas en que se concentran más estas características como las esquinas de los ojos, nariz y boca, las áreas del mentón y la frente [9] [16].

En trabajos previos en el estado del arte son definidas tres tipos de características, como globales, locales o hibridas. Cambios desde el nacimiento hasta la edad adulta (crecimiento craneofacial) son codificados como características globales, mientras que los cambios de la piel desde la edad adulta hasta la vejez se codifican por características locales. Por lo tanto, características globales y locales se combinan para formar una completa representación de envejecimiento facial [9].

Para la extracción de características globales son usadas los modelos de apariencia, entre estos el más utilizado en varios trabajos son los Modelos de Apariencia Activa (AAM) [9] [16] [17], que codifican la información de la apariencia y forma [18] a partir de las anotaciones de las estructuras del rostro [19] [20] [21]. El motivo de utilizar también las características locales es para mejorar el problema que se presenta con AAM, que se pierde información de textura y piel, que resulta ser de vital importancia para la clasificación de la edad [9]. En este sentido, los métodos más utilizados para extracción de características locales son Local Binary partterns (LBP), el conjunto de filtros Gabor Wavelets [9] [16] y la transformada discreta Wavelet (WDT) [17], De esta forma se obtiene una robusta caracterización de la textura de la zona del rostro. Los LBP son utilizados en varios trabajos para la descripción de texturas, en donde se definen con diferentes parámetros de números de píxeles y de radio [22] [23], incluso con el histograma como vector de características [24] [25]. Las *Gabor Wavelets* permiten caracterizar las arrugas del rostro al variar la escala y la orientación del filtro. En el estado del arte el mejor resultado han sido con 5 escalas y 8 orientaciones [26] [27], creando el vector de características a partir de la respuesta en magnitud y la real. Mientras que la transformada discreta Wavelet es una herramienta para la compresión de imágenes, realce de bordes, análisis de textura y eliminación de ruido [28] [29].

En forma de obtener una representación completa de la cara, se realiza la fusión de las características globales y locales por medio de la concatenación de cada uno de los vectores (características hibridas) [9]. Con la matriz de características obtenida, en algunos trabajos implementan una reducción de dimensionalidad a la matriz con *Principal Component Analysis* (PCA) para reducir costos computacionales y mejorar el rendimiento [11] [17]. PCA es utilizado como selector de las características más relevantes, al reducir la dimensionalidad de la matriz de características, al solo dejar las que representan la mayor variabilidad [30] [31] [32].

1.3.3. Clasificación

Finalmente en la última etapa de clasificación tiene como objetivo encontrar las fronteras que permitan separar a cada una de las clases, una de las tareas generalmente implica separar los datos de las características de cada una de las imágenes (ejemplos o muestras) en conjuntos de entrenamiento y prueba. Cada instancia en el conjunto de entrenamiento contiene una etiqueta que especifica a que clase corresponde (y_i) y las características que lo definen (x_i). Es así que con este conjunto se entrena el clasificador y se obtiene el modelo que define a las fronteras que separan cada clase. Con este modelo se predice las etiquetas para el conjunto de prueba solo con sus vectores de características X.

Entre los clasificadores más populares en este tipo de tareas están: *Support Vector Machine* (SVM) que es una técnica de clasificación en aprendizaje supervisado, con un buen

rendimiento en las tareas de clasificación [33] [34] y utilizado en varios trabajos de estimación de edad [8] [19] [20] y en [9] implementan un clasificador jerárquico en el cual un SVM determina el grupo de edad y para cada grupo se tiene un SVR que determina el valor de la edad. Otro enfoque de clasificación es el modelo bayesiano en el que se basa la técnica de Procesos Gaussianos (GP) en los cuales además de la etiqueta de clasificación, proporciona una medida de incertidumbre [35] [36] [37] [38]. En [35] [37] implementan una variante de procesos gaussianos llamada WGP (*Warper Gaussian Process*) para múltiples clases y ven la estimación como un problema de regresión. En [35] implementan el clasificador jerárquico en el cual un GPC multi-clase clasifica en diferentes grupos de edad y luego con WGP de regresión modela para cada grupo la edad específica.

1.4. Marco Teórico

A continuación se detallan los conceptos teóricos de cada uno de los descriptores mencionados en el estado del arte y que son implementados. También la técnica de reducción de dimensionalidad *Principal Component Analysis* (PCA), y cada uno de los clasificadores implementados en este trabajo.

Modelo de apariencia activa (AAM)

AAM es un modelo estadístico ampliamente usado para el modelamiento facial y la extracción de características [9] [18] [19] [20], en donde las variaciones de forma y textura se codifican en conjuntos de datos representativos. Estos modelos de apariencia son generados por la combinación de un modelo de forma activa (ASM) que representa la estructura facial y un modelo de textura que representa el patrón de intensidades de los píxeles (textura de la piel) [20]. En [9] utilizan el análisis de componentes principales (PCA) sobre los datos de forma y textura para construir un modelo facial paramétrico.

Los puntos de referencia o los "*facial landmark*" como son conocidos en el estado del arte, son los puntos o anotaciones que indican las coordenadas en una imagen del rostro de las zonas de interés como el contorno de la cara, los ojos, los labios, las cejas y la nariz. Es así que dada una imagen facial y los puntos de referencia, se genera el modelo estadístico de la forma mediante PCA [9] [20].

Una forma se describe como un vector de coordenadas desde los puntos de referencia. Estas formas deben estar alineadas con un marco común para analizar las variaciones de forma en el conjunto de datos por su variación utilizando PCA [9]. Sea un conjunto de imágenes $I = [I_1, I_2, ..., I_N]$, donde N es el número total de imágenes, con puntos de referencia como

$x = [x_1, x_2, ..., x_N]$. Cualquier vector de forma x en el conjunto de entrenamiento se puede representar como en la ecuación 1.1 [17] [21].

$$x \approx \bar{x} + V_s b_s \tag{1.1}$$

Donde $\bar{x}$ es la forma media, V_s contiene los vectores propios de los valores propios más grandes (λ_s) y b_s representa pesos o parámetros del modelo de forma [21]. Al reescribir la ecuación 1.1, es posible calcular los parámetros del modelo de forma correspondientes a una nueva imagen [17].

$$b_s = V_s^T(x - \bar{x}) \tag{1.2}$$

Similar al modelo de forma, el modelado de textura es calculado por medio del parche libre de forma (*shape free patch*) que se obtiene deformando las imágenes de cada conjunto de entrenamiento en la forma media (*shape mean*) [17], un ejemplo de la imagen original con los puntos de referencia y el *shape free patch* se visualiza en la figura 1.1. Sea $g = [g_1, g_2, ..., g_N]$ los vectores libres de forma de todas las imágenes. Donde bg es el parámetro de peso o modelo de nivel de gris, V_g es el vector propio y g es el vector medio de nivel de gris [21].

$$b_g = V_g^T(g - \bar{g}) \tag{1.3}$$

Figura 1.1: Ejemplo de *shape free patch* y de la imagen original con los puntos de referencia. (Tomado de: https://www.groundai.com/project/face-image-analysis-using-aam-gabor-lbp-and-wd-features-for-gender-age-expression-and-ethnicity-classification).

Ya con los parámetros de los modelos de forma b_s y de textura b_g (nivel de gris) son combinados para obtener el parámetro del modelo de apariencia. El vector de parámetros de apariencia b_{sg} se obtiene a partir de 1.4, en donde W_s es una matriz diagonal de pesos para

cada parámetro de forma, necesaria para hacer proporcionales a b_s y b_g y poder compararlos debido a que tienen unidades diferentes [17] [21].

$$b_{sg} = \begin{pmatrix} W_s b_s \\ b_g \end{pmatrix} \tag{1.4}$$

$$b_{sg} = Q_c \tag{1.5}$$

$$Q = \begin{pmatrix} Q_s \\ Q_g \end{pmatrix}$$

PCA se aplica en el vector de parámetros combinados y se calcula el parámetro de apariencia c que controla tanto la forma como la textura del modelo. Al variar c, es posible lograr cambios tanto en la forma como en la textura [17]. Estos parámetros combinados son utilizados como características globales para capturar la información de la forma y la apariencia relacionada con la edad [9] [17].

Patrones Binarios Locales (LBP)

Una característica local (*Local Feature*) es una propiedad que describe a un píxel con relación a su vecindad (píxeles circundantes). Esta característica puede presentar formas específicas con una estructura propia, tales como puntos (*blobs*), contornos (*edges*) o cualquier otra estructura que se adapte a formas habituales. A la unión de varias características locales se le denomina un patron de descripción o descriptor.

Los patrones binarios locales son un descriptor de textura desarrollado por el trabajo de Ojala et [22]; este descriptor de textura toma el valor central de un conjunto de píxeles y lo compara con cada uno de sus vecinos y considera el resultado como un número binario, esta operación posee una capacidad discriminativa alta y debido a lo simple que es a nivel computacional permite llevarla a cabo reiteradamente; además es invariante ante los cambios monotónicos al nivel gris causados por las variaciones de iluminación y traslación; por ello es de uso popular en distintas aplicaciones sobre todo cuando se lleva a cabo en tiempo real [23] [24] [25].

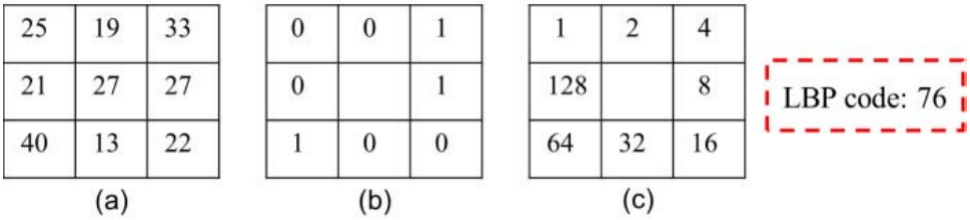


Figura 1.2: (a) Ejemplo de un bloque local; (b) código binario de vencindad entre los píxeles; (c) formato de transformación de código binario a decimal. (Tomado de: https://ieeexplore-ieee-org.ezproxy.utp.edu.co/stamp/stamp.jsp?tp=&arnumber=7243324).

Donde g_c es el valor gris del píxel central y g_p representa el valor gris de un píxel vecino en un círculo de radio R, y P es el número total de vecinos muestreados, como se describe visualmente en la Figura 1.2. En cambio, los píxeles vecinos que no caen en las posiciones enteras se estiman por interpolación bilineal. La principal ventaja de este proceso es que permite capturar detalles extremadamente finos en la imagen. Sin embargo hacerlos a una escala muy pequeña también es el mayor inconveniente del algoritmo; ya que este posee una escala fija de 3×3.

Para manejar esto, Ojala et al. Propusieron una extensión a la implementación original de LBP. Para manejar tamaños de píxeles vecinos variables, se introdujeron dos parámetros: El número de puntos P en una vecindad simétrica circular a considerar (eliminando así la dependencia de una vecindad cuadrada). El radio del círculo R , permite dar diferentes escalas. A continuación se muestra una visualización de estos parámetros [23] [25]:

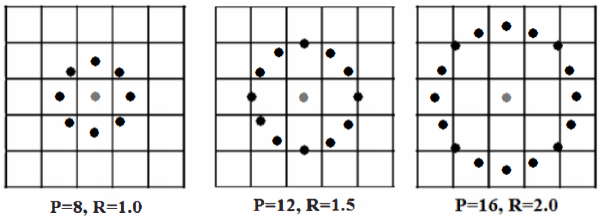


Figura 1.3: Operador LBP Multiescala. (Tomado de: https://ieeexplore-ieee-org.ezproxy.utp.edu.co/stamp/stamp.jsp?tp=&arnumber=7515835).

De forma que la representación de LBP está dada por la siguiente expresión:

$$LBP_{P,R}(i,j) = \sum_{p=0}^{P-1} s(g_p - g_c)2^p \tag{1.6}$$

$$s(x) = \begin{cases} 1, & \text{x} \geq 0 \\ 0, & \text{x} < 0 \end{cases}$$

Con esta nueva expansión de proceso de LBP se tiene en consideración un concepto que se denomina **uniformidad**. De manera que un conjunto de píxeles es uniforme cuando solo existen como máximo dos transiciones *0-1* ó *1-0* en él, esta cantidad se asocia directamente al número de puntos p. Por lo tanto para una cantidad p existe un conjunto de $p+1$ patrones uniformes, esto es vital importancia para poder extraer el vector de características resultante.

Gabor Wavelet (GW)

Una característica primordial para determinar la edad de una persona son las arrugas faciales, ya que estas dependen de factores como la dirección del músculo, la frecuencia de uso, el efecto de la gravedad. Es por esta razón el interés en la extracción de características locales. Las características locales en imágenes faciales son más robustas a las distorsiones como la pose, la iluminación, cabellos, sombras o bigotes, por lo tanto en varios trabajos implementan para la detección de arrugas los filtros de Gabor Wavelet, que permiten obtener estas características con resistencia al ruido [16] [17].

Las Gabor Wavelet es un filtro utilizado en aplicaciones de procesamiento de imagen para la detección de bordes, análisis de texturas y para la extracción de características. Son adecuados para problemas de segmentación ya que poseen propiedades de localización tanto en el dominio espacial como en el de la frecuencia [16], y lo hacen un buen descriptor de arrugas al variar las orientaciones y escalas del kernel. La Gabor Wavelet se define de la siguiente manera [9] [17]:

$$\psi_{\mu,\nu}(z) = \frac{||k_{\mu,\nu}||^2}{\sigma^2} e^{-\frac{||k_{\mu,\nu}||^2 ||z||^2}{2\sigma^2}} [e^{ik_{\mu,\nu}z} - e^{-\frac{\sigma^2}{2}}] \tag{1.7}$$

Donde μ y ν definen la orientación y la escala de los kernels de Gabor, el vector de onda (*wavelet*) $k_{\mu,\nu}$, se define de la siguiente manera:

$$k_{\mu,\nu} = k_\nu e^{i\phi_\mu}$$

$$k_\nu = k_{max}/f^\nu \, and \, \phi_\mu = \pi\mu/8$$

Donde k_{max} es la frecuencia máxima de la onda, f es definida como factor de separación entre núcleos en el dominio de frecuencia y $z = (x, y)$. En trabajos previos los parámetros

más utilizados para la extracción de características faciales son $\sigma = 2\pi, k_{max} = \frac{\pi}{2}$ y $f = \sqrt{2}$ y un tamaño del kernel de 32×32 píxeles [9] [17] [26] [27].

Los kernels de Gabor en la ecuación 1.7 son todos similares, ya que pueden generarse a partir del filtro, la wavelet madre, variando solo el escalado y la rotación a través del vector de onda Kuv. Cada kernel es producto de una envolvente gaussiana y una onda plana compleja [26]. Para extraer características relevantes de una imagen facial, se requiere un conjunto de filtros de Gabor a diferentes escalas (frecuencias) v y orientaciones u. En el estado del arte se ha demostrado que el mejor rendimiento se ha obtenido con un banco de filtros con 5 escalas $v = [0, 1, ..., 4]$ y a 8 orientaciones $u = [0, 1, ..., 7]$ [17] [26] [27].

Al usar un conjunto de 40 *wavelets* (es decir, ocho orientaciones y cinco escalas), es posible extraer una gran cantidad de información relacionada con las intensidades y orientaciones de las arrugas [9]. El resultado contiene las características faciales más importantes, como los ojos, los bordes de la boca y la nariz, así como lunares, hoyuelos y cicatrices. Por lo tanto una vez se tienen el banco de filtros, se realiza para cada filtro la convolución con la imagen para obtener la respuesta tanto la real como en magnitud, es decir se obtienen 80 respuestas para una sola imagen.

En el estado del arte toman estas 80 respuestas real y de magnitud y las concatenan en un solo vector, lo cual da como resultado una alta dimensionalidad, por lo que proceden a realizar un *downsample* por un factor r, y normalizan el vector a media cero y varianza unitaria [26] [27]. Mientras que en otros trabajos de la respuesta en magnitud de la convolución de cada filtro con la imagen, a cada respuesta obtienen la media y la varianza, porque la media y la varianza de la magnitud representan tanto la fuerza como la cantidad de arrugas [9] [16]. Concatenando cada medida para finalmente obtener un vector de características de 1×80 columnas para una imagen.

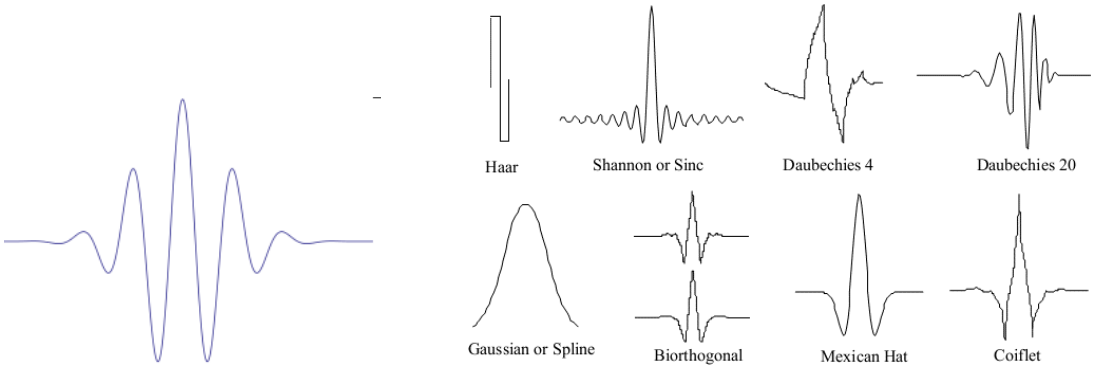
Transformación Wavelet (TW)

La transformada Wavelet es una herramienta que permite realizar el análisis de señales no estacionarias así como también empleada para la compresión de imágenes y reconocimiento de patrones; por otro lado permite tomar ciertas regiones de la imagen en donde realmente no importa la forma si no la existencia de las mismas regiones; es decir que la Transformada Wavelet provee análisis de multiresolución con ventanas o kernels dilatados, de manera que se toman ciertas componentes de frecuencia para ser analizadas, de forma que para un mayor rango se realiza usando ventanas angostas y el análisis de las frecuencias de menor rango se hace utilizando ventanas anchas, todo con una resolución conforme a su escala [28].

Para una representación en (1D) las señales poseen cambios lentos o vibraciones debido a frecuencias transitorias; para una imagen (2D) se presenta como un cambio brusco en una

región específica cuando se encuentran bordes o se cambia el contraste. Estos cambios bruscos son precisamente lo que hace denotar ciertos patrones que para el caso son de gran aporte para la toma y análisis de información [29].

Dado que la transformada de Fourier permite construir una señal a partir de sumas de ondas sinusoidales que no se limitan en tiempo ni espacio, es lo que precisamente no permite representar eficientemente cambios repentinos en la señal que se desea recrear y para ello se debe emplear una nueva transformación la cual esté localizada en tiempo y frecuencia. Mientras que las señales sinusoidales se extienden hasta el infinito, las wavelets existen por un periodo de tiempo finito. Además, estas difieren en tamaño y forma lo que otorga una fortaleza clave para el análisis dada la necesidad.



(a) Wavelet Finita. (b) Diferentes kernels Wavelet.

Figura 1.4: Filtros de la transformada Wavelet. (Tomado de: https://www.researchgate.net/figure/Examples-of-types-of-wavelets-7_fig2_303144392).

De forma más general la transformada Wavelet de una función $f(t)$ es una descomposición generando varias funciones $\Psi_{s,\tau}(t)$ que forman un conjunto denominado "Wavelets", definiéndose de la siguiente forma:

$$W_f(s,\tau) = \int f(t)\Psi^*_{s,\tau}(t)\ dt \tag{1.8}$$

Donde s es el factor de escala, y τ es el factor de traslación.

La salida de la transformación produce el mismo número de coeficientes que la longitud de la señal de entrada. Este proceso se realiza comparando una señal con un banco de filtros de tasa múltiple discreto.

Selección de características PCA

PCA es un método de transformación lineal, tiene como objetivos encontrar algún patrón entre los datos y detectar la correlación entre las variables. PCA permite reducir la dimensionalidad de los datos, la cual solo tiene sentido si hay una gran correlación entre estos. En otras palabras, PCA encuentra las direcciones de máxima variación (Componentes principales - maximizan la varianza) de los datos de alta dimensionalidad y los proyecta a un sub-espacio de dimensionalidad más pequeño, conservando la mayor parte de la información relevante [30].

El objetivo deseado de PCA es reducir la dimensionalidad de los datos ($k < d$, donde k es baja dimensionalidad, y d es la alta dimensionalidad o la original del conjunto de datos) para aumentar la eficiencia computacional reteniendo la mayor parte de la información.

En el conjunto de datos entregado, PCA encuentra un nuevo sistema de coordenadas a partir del antiguo, solo con la traslación y la rotación. PCA mueve el centro del eje de coordenadas (x, y) al centro de los datos, y deja como nuevo eje $x^{'}$ al eje principal de variación (es el eje de mayor variación en relación con todos los datos), el nuevo eje $y^{'}$ es una dirección de variación ortogonal a $x^{'}$ de menos importancia [31].

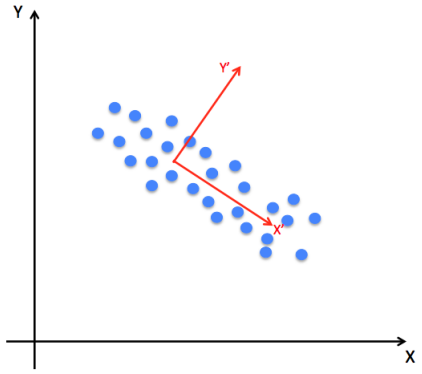


Figura 1.5: Nuevo eje de coordenadas con PCA. (Tomado de: https://medium.com/machine-learning-bites/machine-learning-unsupervised-learning-principal-component-analysis-8f7ad311027e).

Cuando se realiza la selección de características con PCA, se detectan las características compuestas o las "*Principal Component*" (PC). Con las PC se trata de encontrar una dirección en los datos en donde se puedan proyectar perdiendo la menor cantidad de información. En otras palabras PC de un conjunto de datos es la dirección que tiene la mayor varianza, significando que contiene la cantidad máxima de información de los datos originales, disminuyendo así la pérdida de información [31].

Cada componente principal PC es ortogonal entre sí, sin superposición, por lo que son una especie de características independientes. Finalmente PCA ayuda a visualizar datos de alta dimensión, reduce el ruido y hace que otros algoritmos de clasificación funcionen mejor al reducir la cantidad de características y a su vez mejorando la eficiencia computacional [31] [32].

Por lo tanto, el procedimiento es encontrar los PC o *eigenvectors* (vectores propios) del conjunto de datos y construir una matriz de proyección. Los *eigenvectors* también traen asociado a ellos los *eigenvalues* (valores propios) que son la magnitud de los vectores propios y explican la varianza de los datos a lo largo de los nuevos ejes de características. Los *eigenvectors* y los *eigenvalues* se obtienen de la descomposición propia de la matriz de covarianza o de correlación, con los datos estandarizados a media cero y varianza unitaria [30].

Una vez se tienen los *eigenvectors* y *eigenvalues*, se tiene que seleccionar que PC son los que contienen mayor varianza e información de los datos. Para esto se ordenan los *eigenvalues* de mayor a menor para elegir el mejor k *eigenvectors* (PC), en donde los *eigenvalues* más bajos se pueden despreciar o eliminar. Se continúa con la creación de la matriz de proyección W utilizada para transformar los datos al nuevo sub-espacio de características, esta matriz es solo concatenando los k vectores propios elegidos [11] [30] .

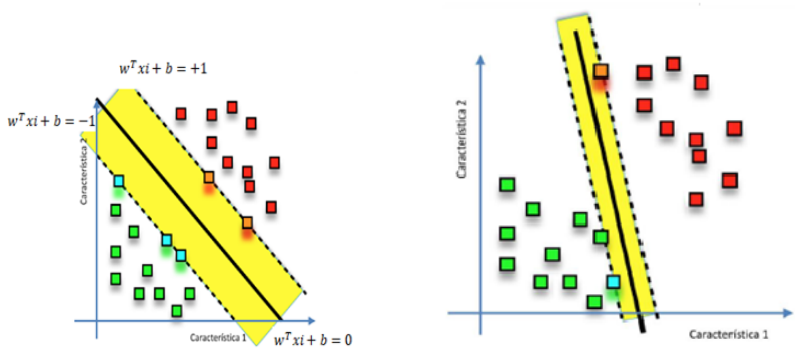
Finalmente se usa la matriz W para transformar el conjunto de datos al nuevo subespacio, se pasa entonces de una alta dimensionalidad d a una dimensionalidad menor k correspondiente a las dimensiones de $W(k < d)$.

Máquina de Sopore Vectorial (SVM)

SVM es un clasificador binario es decir permite distinguir entre dos clases. Es un clasificador lineal, por lo tanto para un espacio de mayor dimensionalidad la frontera de decisión corresponde a un hiperplano; como no existe una única manera de trazar un hiperplano para separar el espacio de características de las dos clases, para SVM este hiperplano se determina de manera que se maximice la distancia entre los ejemplos de cada clase más cercanas entre sí, a lo que se denomina como el margen entre las 2 clases [33] [34]. Estos ejemplos (los casos más difíciles de separar) conforman los Vectores de soporte (*Support Vector*), en la figura 1.6 se logra visualizar dos ejemplos de los vectores de soporte (SV) y el margen que forman entre las dos clases.

SVM es un modelo discriminativo en donde se estima por medio de un conjunto de datos de entrenamiento, tomando un número limitado y determinado de muestras SV para cada una de las clases. Los SV lo conforman las muestras más cercanas a la frontera de clasificación y por lo tanto las más difíciles de clasificar, la elección de estos vectores de soporte es que deben

cumplir con la condición de que la región entre los SV sea la más amplia posible (margen máximo) y que dentro de esta región no contenga ninguna muestra [33] [34].



(a) Vectores de soporte para cada clase con sus muestras y el hiperplano intermedio entre estos vectores es la solución de SVM (la frontera de clasificación).

(b) Una elección diferente de los vectores de soporte dan un margen menor y un hiperplano diferente.

Figura 1.6: Hiperplanos. (Tomado de: Universidad Autónoma de Barcelona, "SVM - conceptos básicos", curso online de detección de objetos, Coursera, https://www.coursera.org/learn/deteccion-objetos).

Dado el conjunto de datos de entrenamiento con N ejemplos y sus respectivas características $X = \{x_i\}^N$ siendo $x_i \in \mathbb{R}^D$ el vector de una imagen y D la dimensión del vector de características, con el conjunto de etiquetas $y = \{y_1, ..., y_N\}^T$ que para las dos clases son $y = (+1, -1)^T$. El hiperplano solución que separa completamente las dos clases está definido por $w^T x_i + b = 0$, este hiperplano solución se obtiene como el hiperplano medio entre los dos hiperplanos de los SV los cuales están definidos de la forma $w^T x_i + b = +1$ para la clase positiva, y $w^T x_i + b = -1$ para la clase negativa. Combinando a estos dos por medio de su etiqueta y_i se obtiene la restricción para que las muestras estén más allá del margen de los SV $(yi(w^T xi + b) \geq 1)$ [20] [33]. Es así que el margen o la distancia entre SV es 1.9:

$$margen = 2\frac{1}{||w||} \tag{1.9}$$

Por lo tanto, la solución de SVM es encontrar el margen máximo entre las dos clases a partir de los vectores de soporte. Entonces el hiperplano o frontera de clasificación se obtiene a partir de un problema de optimización, en la cual se máximiza la distancia entre los SV

(vectores de soporte) de las dos clases, es decir se encuentra el mejor par de SV que contenga el mayor margen de separación [34]. Cualquier otra elección de los vectores de soporte se genera un margen menor, como se aprecia en la figura 1.6(b).

Es así que maximizar el margen es un problema de minimización del inverso del cuadrado del margen que con la restricción anterior se vuelve un problema de optimización cuadrática que se resuelve utilizando la formulación dual (Condiciones de *Karush-Kuhn-Tucker*. KKT) [33]. Para esto se construye el Lagrangiano combinando la función a optimizar y las restricciones multiplicadas por los multiplicadores de Lagrange α 1.10, realizando la minimización de L (derivadas parciales) con respecto a w,b, se obtiene un nuevo problema de optimización que es la solución final de SVM denominado SVM dual 1.11, sujeto a una nueva restricción [20] [33] [34].

$$L(w, b, \alpha_i) = \frac{1}{2} w^T W - \sum \alpha_i [y_i (w^T x_i - b) - 1] \tag{1.10}$$

$$\text{máx} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j (x_i, x_j)$$

$$\sum_{i=1}^{N} y_i \alpha_i = 0 \tag{1.11}$$

La resolución de 1.11 se obtiene el valor óptimo de w definido $w = \sum \alpha_i y_i x_i$ la cual es una combinación lineal con las muestras de entrenamiento x_i y que para los valores de α >0 corresponden a las muestras de los vectores de soporte [34].

En general, debido a que la clasificación en SVM depende únicamente de los vectores de soporte, esto genera ciertas características particulares: *i*) Si se desplazan las muestras asociadas a los SV (cambian la posición de los SV) evidentemente cambia el margen y la solución de la frontera. *ii*) Como solo depende de las muestras difíciles, esto proporciona cierta robustez estadística y reduce el sobre-entrenamiento (*overfitting*) al no depender de todo el conjunto de entrenamiento, añadiendo cierta generalización. *iii*) Al contrario, si se cambian las muestras de entrenamiento, pero los vectores de soportes siguen siendo los mismos entonces la solución del hiperplano seguirá siendo la misma [34].

Conjuntos no linealmente separables:

En algunos casos se puede encontrar que por la distribución de cada uno de los ejemplos en el espacio de características, la solución sea no lineal y para que SVM sea efectivo en esta situación, se tiene las alternativas por margen suave, o por la técnica de *Kernel trick* [34].

- Margen suave: Se trata de relajar la condición del margen con una tolerancia al permitir que algunas muestras estén dentro de la región de los SV, le otorga tolerancia a errores de clasificación y que sea robusto al ruido asociado a las muestras [20] [34].
- Kernel trick: Se trabaja el conjunto no lineal por medio de una transformación o mapeo a un espacio de características lineales de mayor dimensionalidad, este mapeo se realiza fácilmente al definir un producto escalar (kernel) [34].

SVM multi-clase:

Para este trabajo se consideran 4 clases de edades para la clasificación, por lo tanto el enfoque binario de SVM no es suficiente y ampliar el proceso de optimización a muchas restricciones no es eficiente y tiene un alto costo computacional. De tal forma se ha encontrado que para problema multi-clase en el estado del arte se han trabajado con varias técnicas, entre las cuales una muy popular es la técnica de uno contra todos (*One vs All*) [33].

Uno contra todos consiste en implementar tanto clasificadores SVM binarios como clases existentes, es decir son necesarios k clasificadores los cuales se encargaran de separar cada clase de las demás. Entonces la clasificación se realiza tomando los datos de características de una clase y etiquetándola como positiva ($y_j = +1$), mientras que para los otros ejemplos de las demás clases se les asigna la etiqueta negativa ($y_j = -1$). De esta manera, para el ejemplo anterior (4 clases), la matriz de descomposición que se genera en esta arquitectura es [33]:

$$D_{0-v-R} = \begin{bmatrix} +1 & -1 & -1 & -1 \\ -1 & +1 & -1 & -1 \\ -1 & -1 & +1 & -1 \\ -1 & -1 & -1 & +1 \end{bmatrix}$$

Procesos Gaussianos (GP)

Los procesos Gausianos (GP) es un poderoso método bayesiano no paramétrico para el aprendizaje supervisado. Proporciona predicciones probabilísticas y permite el uso de enfoques bayesianos estándar para la selección del modelo [35]. Su mayor ventaja es que proporcionan una estimación confiable de su propia medida de incertidumbre, en la cual puede ser representada como un conjunto de posibles resultados y sus respectivas probabilidades, lo que se denomina distribución de probabilidad un concepto clave para los Procesos Gaussianos [36].

Un proceso Gaussiano (GP) permite describir las distribuciones de probabilidad sobre las funciones. A partir del conjunto de datos con N ejemplos y los puntos de los datos de

entrenamiento $X = \{x_i\}^N$, con el correspondiente conjunto de etiquetas verdaderas para cada clase $y = \{y_1, ..., y_N\}^T$, en donde $x_i \in \mathbb{R}^D$ corresponde al vector de características para una muestra (imagen facial), D es el número de características o la dimensión del vector de características. Lo anterior corresponde en general a lo que se tiene en un problema de clasificación y lo que interesa es obtener la etiqueta de predicción de un conjunto de datos de prueba x_{test} para la clasificación [35].

Considerando primero una clasificación binaria (de dos clases) en donde las etiquetas de las clases son $y = (+1, -1)^T$. Con GP la relación entre el conjunto de datos de entrada X y las etiquetas de las clases y se modela con la función f con ruido adictivo gaussiano $y_i = f(x_i)+\epsilon$, en donde $\epsilon = N(0, \sigma^2)$ es el ruido gaussiano con media cero y varianza σ^2 [35]. El objetivo es predecir $p(y = c)$ la clase a la que pertenece el nuevo vector de datos X, es decir se estima $p(y = 1|x)$ ó $p(y = -1|x)$, la etiqueta se asigna a la clase con mayor probabilidad [36].

Entonces para el modelo de GP la función de distribución es definida $f|X \sim N(0_n, K)$ donde 0_n denota un vector de ceros de $nx1$, la $N(\mu, \sum)$ es la distribución gaussiana para múltiples variables aleatorias con media μ y la matriz de covarianza $\sum$ (correlación entre variables), y K es la matriz kernel parametrizada por θ [38]. La probabilidad de cada punto de datos se define con la función con ruido gaussiano $y = f(x_i) + \epsilon$ con $(N(0, \sigma^2))$, o expresándolo en forma de vector $y|f \sim N(f, \sigma^2 I_n)$ en donde $y = [y1, ...y_N]^T$ y I_n es la matriz identidad de $n \times n$ [35] [38].

Con lo anterior, GP es un método de kernel Bayesiano con la ventaja de que sus parámetros como θ y σ (hiperparámetros) se aprenden de los datos sin la necesidad de métodos de selección como la validación cruzada. En la estadística Bayesiana generalmente se usa para aprender los parámetros del modelo la probabilidad marginal (*marginal likelihood*) [35] [38]. La función de la probabilidad marginal es definida en 1.12:

$$p(y|\mathbf{X}) = \int p(y|f)p(f|\mathbf{X})df = N(0_n, \mathbf{K} + \sigma^2 I_n) \tag{1.12}$$

Para $-log(p(y|X))$ es decir la probabilidad negativa de log-likelihood (l) se expresa en 1.13. Después es usado el método del gradiente para maximizar l y para estimar los valores óptimos de θ y σ.

$$l = \frac{1}{2}[y^T(\mathbf{K} + \sigma^2 I_n)^{-1}y + \ln|\mathbf{K} + \sigma^2 I_n|] \tag{1.13}$$

Para los modelos de clasificación, donde los objetivos son etiquetas de clase discretas, la probabilidad gaussiana es inapropiada. Por lo tanto en [37] sugieren trabajar con métodos de inferencia aproximada para la clasificación o con métodos numéricos [35]. El método se conoce como clasificación de mínimos cuadrados en [37], aquí se trata la clasificación como

un problema de regresión y las etiquetas de clase como función de un valor real, haciendo la solución analíticamente manejable para las predicciones [35]. De acuerdo con lo anterior, los hiperparámetros optimizados se utilizan para calcular la distribución de predicción para la etiqueta de clase (valor objetivo) de un conjunto de datos de prueba x_{test}. La etiqueta de clase correspondiente a los datos de prueba x_{test} se determina a partir del signo de 1.14 siendo esta la media μ de la distribución Gaussiana. Tenga en cuenta que, junto con la etiqueta de clase, también se obtiene la incertidumbre de predicción [35] [38] en 1.15 siendo esta la varianza θ:

$$\mu_* = (\mathbf{k})^T(\mathbf{K} + \sigma^2 I_n)^{-1} \tag{1.14}$$

$$\sigma^2 = k(x_*, x_*) + \sigma^2 - \mathbf{k_*}^T \mathbf{K_T}^{-1} \mathbf{k_*}^T \tag{1.15}$$

Los procesos gaussianos definen una distribución sobre las funciones, es decir, GP es una función y no un solo valor. El GP se describe completamente por su media y la función de covarianza (también llamada kernel) [37]. Los GP son no paramétricos (a pesar de los hiperparametros del núcleo) esto significa que también van a necesitar tener en cuenta los datos de entrenamiento cada vez que se haga la predicción. Con la elección del kernel permite que los GP tengan propiedades de generalización más allá de los datos de entrenamiento dada la propiedad de incertidumbre [36].

Para la clasificación multiclase la mayoría de los clasificadores con múltiples etiquetas tratan el problema como una clasificación binaria de uno contra todos. En la clasificación de uno contra todos se crean clasificadores binarios M como clases existentes, es así que para la clase m^{th} se entrena un clasificador binario GP considerando las etiquetas de esa clase como positivos ($y = +1$) y para las demás clase se consideran negativos ($y = -1$). En la clasificación multiclase, la etiqueta de clase predicha final es aquella que logra la media posterior predictiva más alta [35], es decir 1.16:

$$y_* = argmax_{m=1,\ldots,M}(\mu_*) \tag{1.16}$$

Regresión Logística Multiclase (MLR)

La regresión logística es un algoritmo de gran relevancia empleado en *Machine Learning* debido a que permite la clasificación de los datos ya sea a pequeña o gran escala, además de otorgarle al sistema, la capacidad de un aprendizaje automático estadístico; tiene la particularidad de ser un método de aprendizaje supervisado [39]. En la figura 1.7 se visualiza en términos generales la estructura de este modelo.

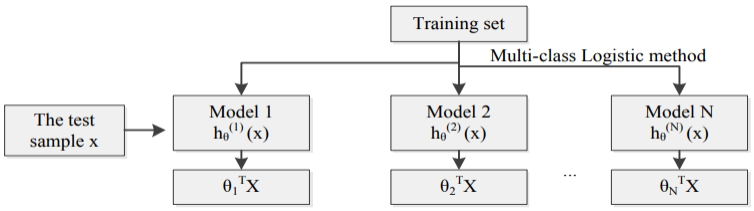


Figura 1.7: Diagrama de flujo de la regresión logística multiclase. (Tomado de: https://ieeexplore-ieee-org.ezproxy.utp.edu.co/stamp/stamp.jsp?tp=&arnumber=8705505).

La regresión logística en su forma más simple emplea una función para representar y predecir el resultado de una variable que normalmente es dicotómica (valor de 1 ó 0) variable dependiente y [40], aunque este algoritmo también posee otros avances los cuales permiten tener más de dos clases para poder modelar diferentes eventos. Este proceso se lleva principalmente a cabo a través de una función *Sigmoide* que permite la no linealización de la regresión, con lo cual permite una mayor generalización del algoritmo [41] [42] [43]. Por lo que en términos generales este modelo predice la probabilidad de éxito o no de algún suceso $P(Y = 1)$ en función de x.

El concepto de regresión se manifiesta en una fórmula matemática que traduce la relación entre variables correlacionadas. Generalmente cuando se quiere poner una variable en función de otra (o de otras), se acude al bien conocido recurso de la regresión lineal (simple o múltiple). Esta función utiliza normalmente el método de mínimos cuadrados y funciona de una forma muy concreta desde el punto de vista aritmético.

Mediante la RL se pretende determinar la probabilidad de que ocurra el hecho en cuestión como función de ciertas variables que se presumen relevantes o influyente. Por lo tanto, consiste en obtener una función logística de las variables independientes que permita clasificar a los individuos en una de las dos subpoblaciones o grupos establecidos por los dos valores de la variable dependiente.

La función logística es aquella que halla, para cada individuo según los valores de una serie de variables (X_i), la probabilidad (p) de que presente el efecto estudiado. Una transformación logarítmica de dicha ecuación, a la que se le llama *logit*, consiste en convertir la probabilidad (p) en *odds*. De aquí surge la ecuación de la regresión logística, que es parecida a la ecuación de la regresión lineal múltiple.

Capítulo 2

Metodología

2.1. Implementación

La metodología implementada para el algoritmo de clasificación de rangos etarios en Python, consistió en primer lugar obtener una base de datos de rostros con etiquetas de las edades de las personas, libre o con licencia académica para poder ser usada en este trabajo. Luego se procede con la creación del algoritmo, iniciando con la etapa de preprocesamiento para preparar todas las imágenes de la base de datos, siguiendo con la etapa de extracción de características para la edad en el rostro empleando las técnicas basadas en características globales y locales, para finalmente a partir de las técnicas de *machine learning* generar los clasificadores para las 4 clases de edad, y observar el poder discriminativo de los descriptores en separar las características de las 4 clases para obtener un buen rendimiento incluso con clasificadores simples.

A continuación se presenta con más detalle la metodología implementada en este trabajo para realizar el algoritmo de clasificación en rangos de edad, además se realiza el procedimiento de entrenamiento y test en la cual contienen las tres etapas anteriores, para la cual las imágenes de la base de datos se dividen en dos grupos, uno de entrenamiento y el otro para el test de los clasificadores.

2.1.1. Base de datos

La base de datos de edad FG-NET contiene 1002 imágenes faciales de 82 Sujetos de múltiples razas con edades que van de 0 a 69 años. La base de datos contiene para una misma persona diferentes fotos en diferentes etapas de la vida o a distintas edades, lo que permite modelar mejor el crecimiento o el envejecimiento del humano. El conjunto de datos no está equilibrado ya que el 50 % de los sujetos tienen entre 0 y 13 años. Las imágenes son en color o en escala de grises con una dimensión promedio de 384×487 píxeles, y la resolución varía de 200 dpi a 1200 dpi con una gran variación de iluminación, pose, expresión facial, desenfoque y oclusiones (por ejemplo, bigote, barba, y anteojos) [44].

La base de datos *MORPH* 2 es un conjunto de datos de imágenes de fotografías con metadatos asociados que dan la edad, el origen étnico y el género de cada sujeto. Contiene 55.608 imágenes faciales para la licencia académica, con aproximadamente tres imágenes de envejecimiento por persona que van de 16 a 77 años. Las imágenes son en color y tienen un fondo gris uniforme. La dimensión promedio es de 400×480 píxeles y la resolución está entre 81 dpi y 96 dpi con cambios en la iluminación, pose, expresión facial y oclusión [45]. En [46] seleccionaron 2530 imágenes faciales para su enfoque de estimación de edad, de tal forma que para este trabajo se hizo uso de este conjunto de imágenes. En la figura 2.1 se visualizan algunas imágenes de las dos bases datos.

(a) FG-NET

(b) MORPH

Figura 2.1: Imágenes de ejemplo de las bases de datos.

Debido a que en *FG-NET* el conjunto de edades no está equilibrado, en donde la mayoría de las imágenes está en el rango de $0-21$ años, dejando pocas imágenes para las edades mayores. Por lo tanto, se selecciona un pequeño conjunto de imágenes de MOPRH para compensar las imágenes faltantes y utilizar este nuevo conjunto combinado para la clasificación.

Los grupos de edades para la clasificación en este trabajo constan de 4 clases, las cuales se determinan a partir de la cantidad de imágenes de las 2 bases de datos, esto es debido a que al realizar el proceso de marginado de cada clase estas terminan descompensadas con respecto a la otras con un porcentaje mayor al 50 % del total y por ello, dado que el nombre de cada imagen contiene la edad de la persona; se realiza un algoritmo que clasifica cada una de las imágenes y las asgina a una carpeta correspondiente a la clase, se llevan a cabo varias iteraciones en la clasifiación de edad con el conjunto de datos y se determina que la primera clase corresponde a los niños que van de $0-13$ años, la clase de jóvenes de $14-21$ años, la clase de adultos de $22-39$ años y la clase de adultos mayores de $40-77$ años. En la tabla 2.1 se visualiza la distribución de las imágenes de la base de datos en los 4 grupos de edades.

Clases	FG-NET	MORPH	Total	Entrenamiento	Test
0 - 13	487	0	487	365	122
14 - 21	232	0	232	174	58
22 - 39	185	76	261	196	65
40 - 77	68	267	335	251	84

Cuadro 2.1: Distribución del conjunto de imágenes en las 4 clases y sus etiquetas.

2.1.2. Lenguaje de programación

Para implementar los algoritmos del sistema de estimación de edad se elige el lenguaje Python, el cual es un lenguaje de programación orientado a objetos, es interpretado, y es muy utilizado para aplicaciones de *machine learning*, además de contar con una gran cantidad de paquetes y códigos Open Source para realizar implementación en diversos temas y entre estos se encuentra el *machine learning*. Python cuenta con las versiones 2.x y 3.x. Para este proyecto se trabaja con la versión 3.5.x o la 3.6.x para la cual están implementada la librería Dlib que se utilizara en este trabajo [47].

Entorno de trabajo

El entorno de trabajo o IDE utilizado es *Pycharm*, desarrollado por la empresa *Jetbrain*. Se utiliza la versión libre que es la *Community*.

Librerías y extensiones

- **Numpy:** Librería Open Source que ofrece un gran conjunto de funciones matemáticas y de algebra lineal, como el poder trabajar con vectores y matrices, es necesaria para trabajar la imagen como un arreglo de píxeles.
- **OpenCV:** Es una librería Open Source de *computer vision* y *machine learning*. Esta librería cuenta con más de 2500 algoritmos optimizados, de los cuales pueden ser utilizados para identificar objetos, clasificar acciones en videos, para detectar y reconocer rostros. Es compatible con los sistemas operativos de Windows, Linux, Mac OS y Android, y esta impementada para C++, Python, Java y Matlab. [48]
- **Imultils:** Este paquete incluye una serie de funciones convenientes para trabajar con *OpenCV*, que ayudan a facilitar tareas como, traslación, rotación, escalado de las imágenes.
- **Dlib:** Esta librería es multiplataforma para subprocesos, operaciones numéricas, *computer vision*, *machine learning*. Específicamente se emplea para la detección de rostros, utiliza un modelo de clasificación basado en HOG (*Histrogram of Oriented Gradients*) y SVM (*Support Vector Machine*). También cuenta con detección de puntos de referencia del rostro, útiles para la alineación del rostro.

Preprocesamiento

La primera etapa del sistema es el preprocesamiento de la imagen, el cual consiste en preparar y estandarizar la imagen para su posterior procesamiento. Esta etapa es de vital importancia para el buen rendimiento de las siguientes etapas de extracción de características y clasificación [11]. El preparar la imagen se refiere a la detección del rostro, descartando las demás zonas que no son de interés y que contienen ruido como el fondo [7]. Mientras que la estandarización corresponde a una serie de transformaciones para dejar el rostro en un marco común, en la cual las mayores variaciones sean características relacionadas con la edad, y no con la posición del rostro, iluminación y ruido. Las transformaciones que se llevan a cabo son la alineación o rotación , el escalado y el recorte de la cara detectada [7] [8] [9].

Por lo tanto, como entrada a esta etapa se tienen cada una de las imágenes (muestras) de la base de datos compuesta, y como salida se crea una nueva carpeta para la base de datos con todas las imágenes preprocesadas. En este sentido los siguientes procedimientos se implementan para cada una de las imágenes de forma iterativa, recorriendo cada imagen para realizar el preprocesamiento, para al final guardarla en la nueva carpeta con el mismo nombre y etiqueta de la imagen.

2.1.3. Detección

Como se ha mencionado anteriormente, en esta fase se realiza la detección del rostro en la imagen eliminando las demás zonas que no son de interés. Con el rostro detectado se procede en esta área reducida a localizar los ojos, reduciendo así el tiempo de ejecución del programa. La detección de los ojos se obtiene a partir de los puntos de referencia (*facial landmarks*) y son de gran importancia para la alineación facial [9]. Todas las imágenes se convierten primero a escala de grises para reducir la influencia de colores inconsistentes, además se reduce el costo computacional en términos de tiempo y uso de memoria [11].

Primeramente se procede a la lectura de la imagen por medio de la librería *OpenCV* con el comando *cv2.imread("ruta/imagen.jpg")*, después es necesario pasar la imagen a escalas de grises para poder aplicar el algoritmo de detección facial [9], por medio del comando *cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)* [49]. El algoritmo de detección facial se implementa con la librería Dlib el cual cuenta con un detector de rostros por medio del clasificador HOG+SVM. Es entonces que se inicializa el objeto para hacer la detección facial *detector=dlib.get_frontal_face_detector()*, a este objeto dlib se le ingresa la imagen en escala de grises y retorna las coordenadas de la ventana de la región en donde se ha detectado un rostro [49].

El detector facial puede encontrar varios rostros en una sola imagen, retornando varias ventanas [10], es así que de forma iterativa va entregando las coordenadas por cada detección.

En el caso de este trabajo que las imágenes de la base de datos solo contiene el rostro de una persona, esta funcionalidad no es necesaria, sin embargo debido a que algunas imágenes de la base de datos tienen una baja resolución con diferente iluminación y color, se encontró que en varios casos además de la detección del rostro se hacen otras detecciones erróneas, generando ventanas en otras áreas de la imagen. Para ello se anexo una validación en la cual se almacena la detección que tenga la ventana más grande la cual corresponde a un rostro, mientras que las otras son solo partes de esta como la nariz, gafas, una parte del pelo o de fondo. Las coordenadas de la ventana corresponden a la posición inicial (x,y) esquina superior izquierda, y las coordenadas (w,h) que son el ancho y el alto de la ventana.

En segundo lugar se realiza la detección de los ojos por medio de los puntos de referencia (*facial landmark*) [7] [9], los cuales se logran implementar con la librería Dlib. Esta librería contiene un modelo pre-entrenado para detectar los 68 puntos, para esto se hace necesario primero inicializar el objeto por medio de *predictor=dlib.shape_predictor("archivo")*, para el cual se debe tener el archivo "**shape_predictor_68_face_landmark.dat**" [49]. El objeto predictor necesita como parámetros de entrada la imagen en escala de grises y la detección del rostro, y retorna como salida el par de coordenadas (x, y) en píxeles de cada uno de los 68 puntos. Es así como se obtiene las coordenadas que generan la ventana para cada ojo, identificando perfectamente el ojo izquierdo y el derecho. Característica importante para la alineación del rostro, el conocer la posición de cada ojo [9] [49]. Ventaja que ofrece la librería Dlib frente a *OpenCV* que también realiza la detección del rostro y los ojos pero no permite identificar o separar cada uno de los ojos.

2.1.4. Alineación

Después de realizar la detección facial se procede a realizar una serie de transformaciones geométricas sobre la imagen dejándola preparada para la correcta extracción de características. Es así que se preparar el rostro por medio de redimensión (tamaño píxeles), alineación del rostro respecto al eje x y el recorte de la zona de interés [8] [9] [10]. Estas transformaciones permiten normalizar y estandarizar la imagen debido a que muchos algoritmos de reconocimiento facial y métodos de aprendizaje automático se benefician de esta normalización [7] [11].

Dada las coordenadas de los ojos (*landmarks*), el objetivo de este procedimiento es deformar y transformar la imagen a un espacio de coordenadas (salida), es así que en todo el conjunto de datos se debe de realizar: *i)* encontrar el ángulo de rotación a partir de las ventanas detectadas de los ojos, *ii)* realizar la rotación para que los ojos se encuentren en una línea horizontal paralela al eje x (ángulo de rotación igual a cero) [7] [9], este ángulo se compensa mediante una transformación afin de rotación (conserva cada una de las proporciones del rostro) [49], *iii)* escalar el rostro de modo que el tamaño sea aproximadamente idéntico.

Para la realización de la alineación del rostro, se parte de la ventana de cada uno de los ojos para calcular el centro de esta, es decir el punto central del ojo, esto se logra obteniendo la media de los puntos de la ventana de cada ojo y almacenando estas coordenadas (x, y) en las variables ($xEyeRcenter$,$yEyeRcenter$) para el ojo derecho y ($xEyeLcenter$, $yEyeLcenter$) para el ojo izquierdo. Luego se calcula la diferencia o distancia entre el centro de cada ojo obteniendo a dx y dy [49]. Se calcula en ángulo de rotación con 2.1.

$$dy = yEyeRcenter - yEyeLcenter, \qquad dx = xEyeRcenter - xEyeLcenter \tag{2.1}$$

$$angle = \tan^{-1} \frac{dy}{dx} - 180$$

Después se calcula el punto central entre los dos ojos (arriba de la nariz), para tomarlo como punto de referencia en la rotación, sencillamente se calcula el promedio entre las coordenadas (x, y) del centro de las ventanas de los ojos 2.2:

$$xEyeCenter = \frac{xEyeRcenter + xEyeLcenter}{2} \tag{2.2}$$

$$yEyeCenter = \frac{yEyeRcenter + yEyeLcenter}{2}$$

A partir de aquí ya se cuenta con los cálculos necesarios para realizar la alineación, es así que por medio de una clase dedicada de Python para alinear las caras es la transformación afín (*warpaffin*). Las transformaciones afines se utilizan para rotar, escalar, traducir, etc. Estos tres requisitos anteriores se realizan en una sola llamada con la función de *OpenCV* *cv2.warpAffin()*. Para aplicar la función en necesario crear la matriz de rotación M. Para el cálculo de esta matriz se utiliza *cv2.getRotationMatrix2D(eyesCenter,angle,scale)*, la cual cuenta como parámetros de entrada [49]:

- **EyesCenter:** Es el punto de referencia de rotación, el cual se toma como el punto medio entre los ojos.
- **angle:** Es el ángulo que se debe girar el rostro para asegurar que los ojos se encuentren a lo largo de la misma línea horizontal (eje x).
- **scale:** Es el porcentaje o factor para escalar la imagen al tamaño deseado (escalar por arriba o por debajo de la imagen), la escala se obtiene con la distancia entre los ojos deseada (en píxeles) y la distancia real en la imagen actual, $scale = \frac{desiredDist}{Dist}$ en donde la distancia real se obtiene $Dist = \sqrt{dx^2 + dy^2}$.

Para finalizar se aplica la transformación con *cv2.warpAffine(image,M,(w,h),flags=cv2.INTER_CUBIC)* la cual se le ingresan como parámetros: *image* corresponde a la imagen original, M es la matriz de rotación, w define el ancho del rostro deseado en términos de píxeles en este caso de 256 píxeles y h define la altura del rostro deseado en 256 píxeles, obteniendo de esta forma una imagen cuadrada de 256×256 píxeles [49]. Al final se obtiene como salida el rostro de la persona alineada con las escalas deseadas (ancho, alto, distancia entre los ojos) [8] [9].

Recorte: Una vez se tiene alineado el rostro, se hace el recorte de la imagen, para este caso se volvió a detectar el rostro obteniendo las coordenadas (x, y, w, h) de la nueva imagen alineada al eje horizontal, con estas coordenadas se forma la ventana de detección de solo el rostro de la persona y se realiza el recorte de esta zona, eliminando el cabello y parte de la barbilla. De esta forma se asegura de que a las siguientes etapas llegue solamente la región de interés (solo el rostro) para realizar la correcta extracción de características [7] [10].

Escalado: El escalado del rostro detectado se aplica con el objetivo de que todas las imágenes a procesar tengan las mismas dimensiones (256×256 píxeles), es así que para algunas imágenes la ventana de detección de rostro se aumenta o disminuye [9] [10] [11]. En este sentido en la figura 2.2 se logra apreciar completamente el diagrama de flujo de la etapa de preprocesamiento.

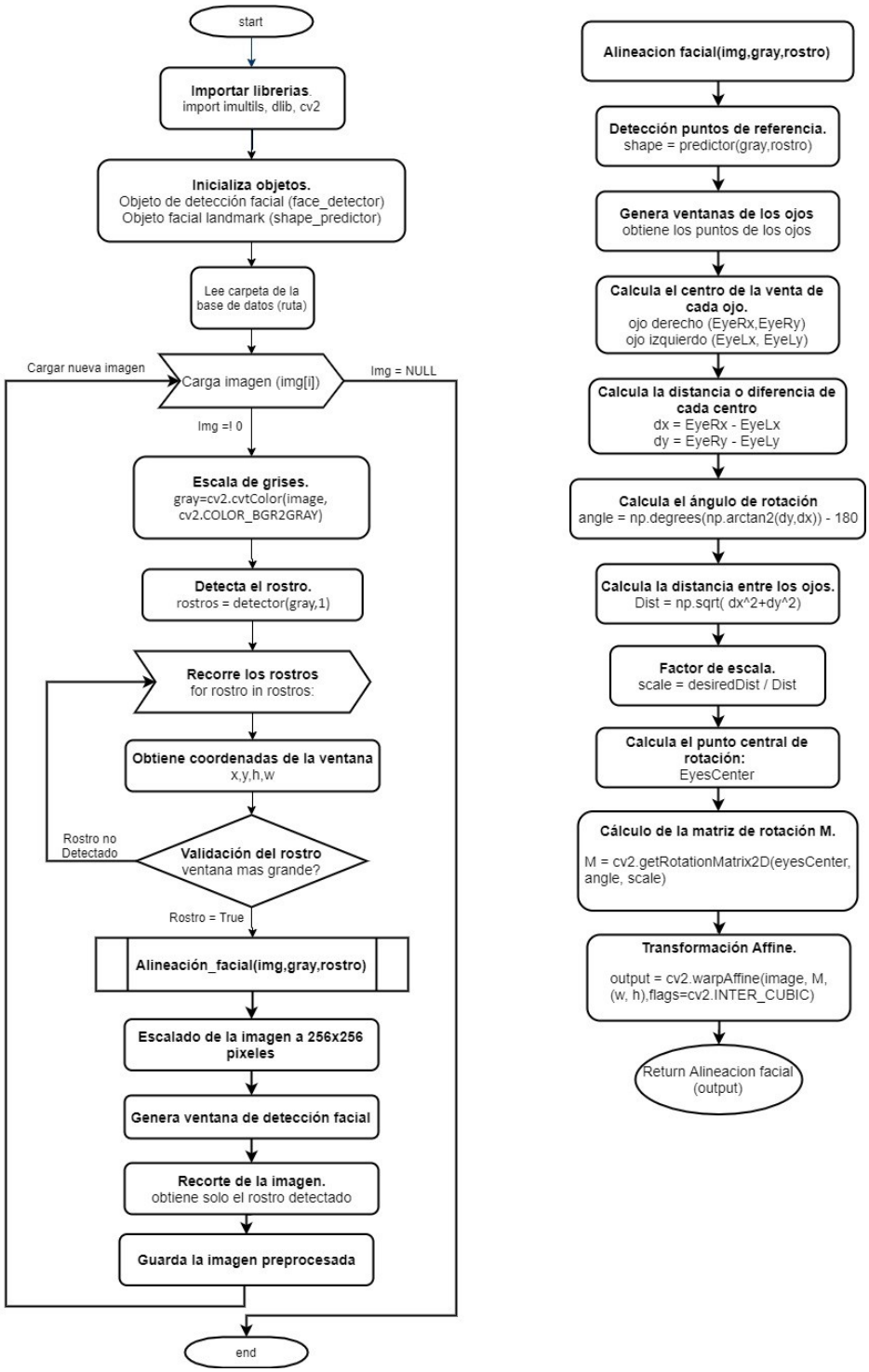


Figura 2.2: Diagrama de flujo de la etapa de preprocesamiento (detección, alineación, escalado y recorte).

2.1.5. Extracción de características faciales

El proceso de extracción de características tiene como objetivo obtener un conjunto de datos (que se conoce como vector de características) a partir de la imagen después de la etapa de preprocesamiento, la idea de la extracción de características es resaltar aspectos del rostro de la persona en la imagen, en este caso características que ayuden a determinar la edad como las arrugas, la textura de la piel, el crecimiento craneofacial. En el análisis del procesamiento facial, existe un diverso número de métodos para la extracción de características con la particularidad de ser robustos al ruido como el cabello, la barba, gafas, sombras, la iluminación o la posición del rostro. En este estudio se implementan dos grupos: las características globales las cuales codifican la forma del rostro, logrando obtener así el crecimiento craneofacial, en este sentido se implementa la técnica AAM el cual modela la forma facial. En segundo lugar están las características locales las cuales codifican la textura de la piel, entre estas se implementan técnicas para detectar bordes y texturas como LBP, los filtros Gabor Wavelet, y la Transformada Wavelet. El otro proceso en esta etapa es la reducción de dimensionalidad del vector de características por medio de la técnica *Principal Component Analisys* (PCA) la cual permite obtener las características con mayor variabilidad.

Modelo de Apariencia Activa (AAM)

Active Appearance Model (AAM) son modelos estadísticos que capturan la variabilidad de la forma y la textura en un vector de características, es decir, produce un modelo de cara paramétrico [20] [21]. Es así como AAM está conformado con un modelo de forma y textura, en donde una forma se describe en un vector de coordenadas (x, y) desde los puntos de referencia (*landmarks*). Sin embargo en [9] muestran como el modelo de textura elimina varias características primordiales para la clasificación de la edad, como las arrugas. Es por esto que en este trabajo no se calcula el modelo de textura de AAM, si no que se implementan las características locales para obtener la información relevante de textura, mientras que con los puntos de referencia se anotan las regiones faciales más relevantes para obtener información sobre la variabilidad del crecimiento craneofacial como fue mencionado en el estado del arte.

Los puntos de referencia tienen como fin anotar y obtener importantes estructuras faciales, en donde los puntos son coordenadas (x, y) en píxeles dentro de una imagen con un rostro. Estos puntos definen importantes regiones como el contorno del rostro, los labios, los ojos o las cejas, características utilizadas para detecciones, alineaciones, transformaciones y en diversas aplicaciones en visión por computador y en la extracción de características como es el caso de AAM [9] [17]. En la figura 2.3 se aprecian cada uno de estos puntos detectados en un rostro, y en la tabla 2.2 se detallan los puntos que conforman cada una de las regiones.

Figura 2.3: Puntos de referencia (facial landmarks). (Tomado de: https://www.pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/).

Regiones Faciales	Puntos de referencia
Contorno de la mandíbula	1 - 17
Ceja izquierda	18 - 22
Ceja derecha	23 - 27
Nariz	28 - 36
Ojo izquierdo	37 - 42
Ojo derecho	43 - 48
Boca	49 - 68

Cuadro 2.2: Distribución de los 68 puntos de referencia para cada una de las regiones faciales.

El objetivo de este descriptor es detectar importantes estructuras faciales que permiten identificar el crecimiento craneofacial, por medio de un modelo de predicción de forma. Los pasos realizados por este es:

- Localizar el rostro en la imagen.
- Detectar las estructuras faciales claves.

Los puntos de referencia (*facial landmark*) como se mencionó en la etapa de preprocesamiento, son obtenidos con la implementación de la librería Dlib el cual contiene el modelo pre-entrenado para detectar los 68 puntos de referencia. Primero es necesario inicializar el objeto por medio *predictor=dlib.shape_predictor(archivo)* este modelo necesita los datos del

archivo "*shape_predictor_68_face_landmarks.dat*" para poder predecir la forma. Luego es importante hacer la detección del rostro y obtener las coordenadas de la ventana facial (x, y, h, w). En segundo lugar en la región detectada se obtienen los puntos de referencia por medio de *shape = predictor(gray,rostro)*, ingresando la imagen en escala de grises y las coordenadas del rostro. Es así que *shape* contiene cada uno de los puntos de referencia en una matriz de dimensiones 68×2, estos puntos son la dupla de coordenadas (x, y) que indican la posición en píxeles en la imagen [49].

Para terminar el vector de características AAM se obtiene al concatenar las coordenadas (x, y) de cada uno de estos puntos, generando un vector de 1×136 características (F_{AAM}) para una sola imagen.

Patrones Binarios Locales (LBP)

LBP es un descriptor popular para la extracción de texturas y reconocimiento de patrones [23] [25], da una representación de la textura local por medio de la comparación de los niveles de grises con los píxeles de los vecinos alrededor. Un beneficio del LBP es que calcula detalles extremadamente finos en la imagen (depende del tamaño de la matriz que define los vecinos). La codificación de la piel para este caso se realiza de forma local donde se calcula una textura con base en la imagen de entrada (rostro), este tipo de representación se obtiene realizando una comparación entre un píxel central y su vecindad circundante; este proceso se puede llevar a través de dos tipos de ventana o kernel.

- Un kernel cuadrado de tamaño fijo, normalmente es de 3×3.
- Un kernel circular variable donde se determina el radio de dicha circunferencia y la cantidad de puntos que se han de tomar dentro del mismo.

El primer paso para poder construir el descriptor es a través de una conversión de la imagen del rostro es una nueva representación, para este proceso se convierte en una imagen en escala de grises; dado que se requiere una mayor descripción del rostro se implementa el kernel variable, asignando a cada parámetro correspondiente r y p valores que permitan abarcan en gran manera regiones de interés; para después calcular el valor LBP para este píxel central, y se almacena en una nueva matriz de las mismas dimensiones de la imagen. Empleando el paquete *scikit image* con el módulo *feature* que permite la implementación de LBP, por medio del comando *lbp = feature.local_binary_pattern(image,numPoints,radius,method="uniform")*, en donde se le ingresan como parámetros la imagen del rostro preprocesada en escala de grises, el número de puntos p a lo largo de la circunferencia, el radio de la circunferencia que define a los vecinos, y como método "uniforme" que indica que se calcula con la rotación y la forma invariante en

escala de grises de LBP uniforme. Y retorna la matriz 2D de salida de código LBP (misma dimensión de la imagen de entrada) [49].

Por lo tanto para poder utilizarla en un vector de características se construye el histograma con la matriz de códigos LBP. Para construir el histograma se realiza la llamada a *(hist,_) = np.histogram(lbp.ravel(),bins=np.arange(0,numPoints + 3),range=(0,numPoints + 2))* en el cual el primer rango es para los patrones uniformes, y el rango adicional es para los patrones no uniformes, que dependen del número de puntos (vecinos selecionados) p. Después se realiza la normalización del histograma para que la suma total sea 1.

Gabor Wavelet

El método Gabor Wavelet es utilizado con el objetivo de obtener las arrugas del rostro. Debido a que las arrugas faciales tienen diferentes orientaciones y tamaños, como por ejemplo las arrugas de la frente son horizontales mientras que las de al lado del ojo o en las mejillas son verticales o con un ángulo diferente, igualmente las arrugas en una persona mayor son más predominantes (escala) que en un joven o adulto [9] [16].

En el estado del arte, la extracción de características faciales son implementados los filtros Gabor con 8 orientaciones (u) y 5 escalas (v) [26] [27]. De acuerdo con lo anterior, al aplicar los filtros con diferentes escalas y orientaciones de kernels en la imagen, en cada respuesta se obtendrán o resaltaran características diferentes de textura dependiendo de la cantidad de arrugas y sus orientaciones [9] [17], es decir que para cada filtro se tiene un valor distintivo en la ubicación espacial de esa característica.

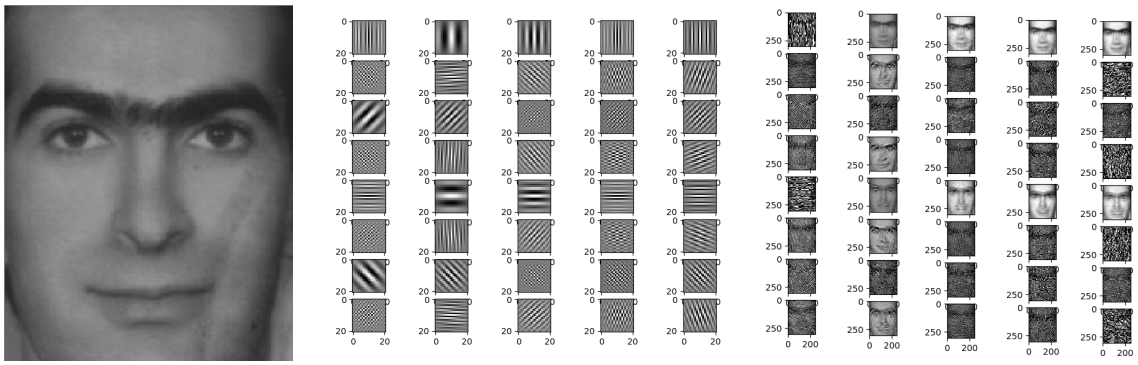
De esta manera se crea un banco de 40 filtros Gabor para la extracción de características. Un filtro Gabor está constituido por una señal sinusoidal compleja con frecuencia y orientación particulares, modulada por una onda gaussiana. Es así que un filtro queda definido por un kernel (matriz) 2D para implementarlo con la imagen y obtener la respuesta por medio de la convolución.

De esta manera para la implementación primero es necesario crear el banco de filtros Gabor, y después realizar la convolución de cada filtro con la imagen. Es así que con la librería OpenCV se crean los kernels con la función *kern = cv2.getGaborKernel((ksize,ksize),sigma,theta,lamda,gamma,psi,ktype=cv2.CV_32F)*, en la cual tiene definida como parámetros de entrada:

- **Ksize:** es el tamaño del kernel en píxeles, es una dupla para formar asi la matriz, el tamaño implementado es de 21×21 píxeles.
- **Sigma:** es la desviación estándar que controla el ancho de banda o el tamaño de la envolvente gaussiana, para este trabajo se implementa $\sigma = 2\pi$ [26].

- **Theta:** representa la orientación en radianes de la función Gabor, se visualiza en la orientación de las rayas de los filtros. Como este parámetro varia para obtener las 8 orientaciones $u = [0, 1, 2, ..., 7]$ esta definido: $\theta = \frac{\pi}{8}u$ [9] [16] [27] [50].
- **Lamda:** Longitud de onda de la sinusoidal, gobierna el anacho de las rayas del filtro. Como este parámetro varía para obtener las 5 escalas $v = [0, 1, ..., 4]$, están definidas por la frecuencia de la señal: $\omega = \frac{\pi}{2f^u} = \frac{\pi}{2\sqrt{2}^u}$ y la longitud de onda es el inverso de la frecuencia $\lambda = \frac{1}{\omega}$ [26] [50].
- **Gamma:** controla la relación de aspecto (altura) de la función $\gamma = 1$ [27].
- **Psi:** Es el desfase de la función sinusoidal $\psi = 0$.

Con cada uno de los filtros creados, se procede a realizar la convolución de cada filtro con la imagen para obtener las respuestas. Esta convolución se implementa con la librería *OpenCV* con la función *cv2.filter2D(img,cv2.CV_8UC3,filters[i])*, en donde se le ingresa como parámetros la imagen (img) preprocesada en escala de grises, y cada uno de los filtros (*filters[i]*) por medio de un for que carga cada uno de los kernels (matriz de 2 dimensiones). Cuando se aplica un filtro Gabor a una imagen por medio de la convolución, proporciona la respuesta más alta en los bordes y en los puntos donde cambia la textura. La respuesta de la convolución es almacenada en una lista y después convertida a tipo *array* para poder indexarla para generar el vector de características. En la figura 2.4 se visualiza los 40 filtros Gabor y sus correspondientes respuestas cuando se aplican a la imagen.



(a) Imagen preprocesada en escala de grises.

(b) Banco de filtros Gabor con 5 escalas y 8 orientaciones.

(c) Las 40 respuestas de aplicar cada uno de los filtros.

Figura 2.4: Entrada y salida del descriptor Gabor Wavelet.

La respuesta de aplicar los filtros Gabor a la imagen proporcionan respuestas más altas en los bordes y en los puntos donde cambia la textura, es decir que para generar el vector de

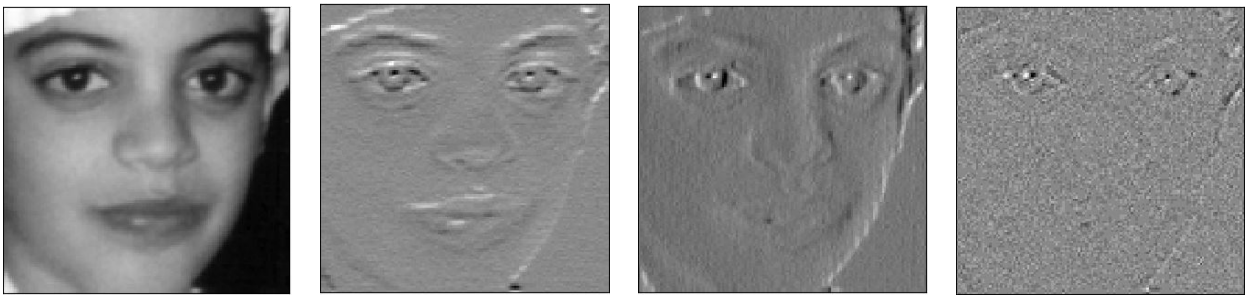
características se extrae una gran cantidad de información relacionada con las intensidades y orientaciones de las arrugas por medio de la media y la varianza de la respuesta de cada filtro [9] [16]. Por lo tanto para una sola imagen preprocesada, a la respuesta de aplicarle un solo filtro se obtienen dos características: la media y la varianza. En este sentido, al aplicarle todo el banco de filtros (40 filtros), al final se construye un vector de características concatenando cada media y varianza de cada respuesta, formando un vector de 1×80 (F_{GW}).

Transformada Wavelet

Esta transformada es empleada con el objetivo de determinar bordes, además de que permite analizar texturas, esto significa que conlleva a obtener características generales o específicas de acuerdo al tamaño de una ventana permitiendo una alta resolución ya sea en el dominio del tiempo o de la frecuencia; lo que con la Transformada de Fourier no se puede hacer, de manera que con esta técnica se resaltan los tres tipos de orientación principales en la imagen preprocesada (rostro), los cuales son verticales, horizontales y diagonales.

De manera que esta transformada utiliza una serie de funciones denominadas *Wavelets* las cuales se pueden escalar de acuerdo al problema en cuestión, que para este caso es la textura de la piel de forma general, es decir todo el rostro. Dado que el *wavelet* esta ubicado con respecto al tiempo, el procedimiento para poder operarla con la señal de entrada es realizando la operación de convolución la cual realiza el desplazamiento en distintos instantes de tiempo permitiendo obtener una respuesta ante dicha onda primaria o principal.

Como se observa en la figura 1.4 existen diferentes tipos de familias *wavelets* las cuales cada una de ellas tiene un efecto diferente sobre la misma imagen, de forma que podemos obtener mejores características variando el tipo de onda a convolucionar. A continuación se visualiza en la figura 2.5 las diferentes orientaciones.



(a) Imagen preprocesada en escala de grises. (b) Detalle horizontal. (c) Detalle vetical. (d) Detalle Diagonal.

Figura 2.5: Tipos de orientación en la transformada wavelet.

2.1.6. Selección de características con PCA

Con la matriz de características X obtenida de la concatenación de las características locales y globales de dimensión $N \times D$ (N "filas" número de ejemplos o imágenes, y D el número de características), el siguiente paso es seleccionar las características más relevantes de la matriz X por medio de la técnica *Principal Component Analisys* (PCA). PCA permite reducir la dimensionalidad de los datos, encontrando las direcciones de máxima variación (Componentes principales PC) [30]. Con el objetivo de reducir el ruido y hacer que otros algoritmos de clasificación funcionen mejor al reducir la cantidad de características, mejorando la eficiencia computacional y evitando el *overfitting* (sobre-entrenamiento) de los datos con los que se entrena el modelo de clasificación, haciéndolo más general [31] [32]. Por lo tanto la nueva cantidad de características dependerá de la elección de los PC. Para seleccionar el número óptimo de componentes principales en este trabajo se realiza un ciclo para ir variando el número de componente PC y evaluar el rendimiento de un clasificador sencillo con cada matriz de entrenamiento Z. Para así al final seleccionar el número de componente PC mínimo a partir del cual el incremento del rendimiento deja de ser sustancial. Primero se expondrá la implementación de PCA obteniendo la matriz Z y luego se explica el procedimiento de selección.

En relación con lo anterior, con la matriz de características X se desea reducir la dimensionalidad de los datos, por lo tanto el procedimiento para implementar PCA es:

- Estandarizar los datos de entrada (Normalización, $media = 0$ y $varianza = 1$).
- Obtener los autovectores y autovalores de la matriz de covarianza.
- Ordenar los autovalores en orden descendente (de mayor a menor) y elegir los autovectores con los autovalores K más grandes (Donde k es el nuevo número de dimensiones del nuevo sub-espacio de características).
- Se construye la matriz de transformación W con los autovectores seleccionados.
- Se transforma la matriz de características X estandarizada al nuevo sub-espacio por medio de W, obteniendo la matriz de características reducidas Z.

Ahora iniciando con la implementación de PCA en lo que respecta a la estandarización de los datos, se emplea el paquete *sklearn.preprocessing* con la función *sc = preprocessing.StandardScaler()* y con el método *sc.fit(X)* se calcula la media del conjunto de datos, para posteriormente aplicar la normalización con *X= sc.transform(X)*, obteniendo el conjunto de datos con $media = 0$ y escalando a la $varianza = 1$. En segundo lugar se calcula la matriz de covarianza (XX^T) con el conjunto de datos estandarizados X y su transpuesta

X^T, la matriz de covarianza es P = *np.matmul(np.transpose(Xpp),Xpp)* en donde (X_{pp} es una copia de X para no modificar los datos originales).

Es así que con la matriz P se calculan los valores *val* (eigenvalues) y vectores propios *vec* (eigenvectors) por medio de *val,vec = np.linalg.eig(P)* y se organizan los valores propios en orden descendente *idx = np.argsort(val)[:: -1]*, *val = val[idx]*, *vec = vec[:,idx]*. Es así como los vectores propios dan las direcciones de máxima variación de los datos, y los valores propios corresponden a su magnitud y explican la varianza de los datos a lo largo de los nuevos ejes de características [30]. Se continúa con la creación de la matriz de proyección W utilizada para transformar los datos al nuevo sub-espacio de características en donde $W = vec$.

Con la matriz de transformación W se calcula la relevancia de cada una de las características *rho_pca = abs(np.dot(W,np.matlib.repmat(val,n,1))).sum(axis=1)*, y se normaliza esta relevancia para que este en valores de 0 a 1, dividiendo primero a *rho_pca* por el mínimo de*rho_pca*, luego por su desviación y por ultimo por su máximo. Con la normalización se organizan las relevancias en orden descendente *idx_pca = np.argsort(rho_pca)[::-1]* y se conservan el número de características (K) o componentes PC seleccionadas (las de mayor relevancia) *idx_pca = idx_pca[0: NumberComp]*. De esta forma se obtiene la matriz de características con reducción de dimensionalidad *Z= X[:,idx_pca]*.

Con el cálculo de reducción de características con PCA para obtener la matriz Z, se procede ahora a determinar el número de PC óptimo para generar la matriz Z. Para esto se crean dos for anidados, los cuales, el primero varia el número de características PC a calcular, desde 1 hasta el número de características originales D en pasos de 5. El segundo for es para entrenar y probar varias veces el modelo de clasificación y obtener un rendimiento promedio, variando el conjunto de datos de entrenamiento y test para dar una mayor generalización.

Enfocando el procedimiento en el segundo for (i_try), este se realiza unas 30 veces para cada una de las componentes PC. En este paso primeramente se divide en 2 la matriz de características X, en un conjunto de datos del 70 % de entrenamiento X_{train} y en un 30 % para test X_{test}, al igual que las etiquetas Y_{train} y Y_{test}, Cabe mencionar que para cada iteración el conjunto de datos *train* y *test* se seleccionan aleatoriamente, razón por la cual se obtiene el rendimiento promedio. Después se aplica PCA como se menciona anteriormente, en la cual con X_{train} se calcula la matriz W y se realiza la transformación a nuevo espacio de características a los dos conjuntos, obteniendo las matrices reducidas Z_{train} y Z_{test} con sus correspondientes etiquetas Y_{train} y Y_{test}.

Ahora se entrena un clasificador simple de regresión logística con el conjunto de datos reducidos Z_{train}, obteniendo los parámetros del modelo en el objeto clf. El entrenamiento de este clasificador se detallara más adelante en la sección de implementación de clasificadores. Luego se realiza la predicción al conjunto de prueba Z_{test} por medio de *y_pred = clf.predict(Ztest)* obteniendo el vector de las etiquetas estimadas

y_{pred}. Se almacena el porcentaje de exactitud de las estimaciones con *Percentage += accuracy_score(Ytest,y_pred)* y se repite el ciclo (*i_try*) variando el conjunto X_{train} y X_{test}, almacenando el rendimiento de cada uno.

Una vez se obtiene la exactitud promedio (30 *veces*) para ese determinado número de características (PC), se incrementa el número de PC para obtener su rendimiento promedio (primer *for*). Generando una gráfica de rendimiento *vs* PC (número de características seleccionadas), para así seleccionar el número de PC óptimo en el cual el incremento del rendimiento deja de ser sustancial. Finalmente con este PC óptimo se trabaja para la siguiente etapa de clasificación, en la cual se crea la matriz con características reducidas Z óptimo con PCA con el número de componentes necesarias, la cual asegura una reducción de características con la mayor variabilidad e información. La gráfica, el número de PC óptimo y las dimensiones de Z son discutidas en el siguiente capítulo de Resultados.

2.1.7. Clasificación

Regresión Logística

Regresión logística es un algoritmo de clasificación supervisado tanto para clasificación binaria como para múltiples clases [42]. La implementación de regresión logística en Python es sencilla gracias al paquete de *Sckit-Learn*. Primero se importa la clase *from sklearn.linear_model import LogisticRegression* con la cual se define el modelo. Después como se mencionó anteriormente se trabaja con la matriz de características reducidas Z para el entrenamiento y validación del modelo [39] [41], es así que se divide la matriz en 70 % para el conjunto de entrenamiento X_{train} y 30 % para la validación y prueba X_{test} y sus correspondientes etiquetas por medio de la función X_{train}, X_{test}, y_{train}, y_{test} = *train_test_split(X,labels,test_size=0.3,random_state=42)*. Con el conjunto de datos divididos para evitar el problema de sobre-generalización, se procede a crear el modelo de regresión con la función *clf = LogisticRegression(random_state=0,solver='lbfgs',multi_class='multinomial',max_iter=300)* y que se establecen como parámetros el solucionador "lbfgs", que es el algoritmo de optimización usado para el problema multiclase multinomial (*multi_class='multinomial'*), y por medio de *max_iter* se establece en 300 iteraciones para que la solución converja. Después se ajusta y se entrena el modelo con los datos y etiquetas de entrenamiento *clf=clf.fit(Xtrain,Ytrain)*.

Con los parámetros y el modelo entrenado en el objeto *clf*, se realiza las pruebas para validar la eficiencia del clasificador. Para esto se realiza la predicción de las etiquetas al conjunto de prueba X_{test} con la función y_{pred} = *clf.predict(Xtest)* y retorna las etiquetas según la clasificación. Es decir que para cada ejemplo o muestra de la matriz con sus correspondientes

características, el clasificador entrega las predicciones o etiquetas que considera que pertenece a determinada clase. Como métrica de medición de desempeño se implementa la matriz de confusión que permite visualizar si las predicciones son reales o se confundieron con otra clase [9]. Para esto se importa el paquete *from sklearn.metrics import confusion_ matrix* y se crea la matriz de confusión basándose en las etiquetas entregadas por el clasificador y_{pred} y las reales y_{test}, con la función *confusion_ matrix(Ytest,y_ pred)*. El desempeño y la matriz de confusión serán mostradas en el capítulo de resultados.

Máquina de soporte vectorial

Posterior a la reducción de características llevadas a cabo a través de el método de PCA se inicia el nuevo proceso el cual es la clasificación de los datos y para ello se realiza el siguiente procedimiento.

Primero se realiza un preprocesamiento de los datos donde se categoriza todo el conjunto de manera que se determinan dos grupos, entrenamiento (*train*) y prueba (*test*). Para este clasificador al igual que en el anterior se determina un 70 % para entrenamiento y 30 % para prueba y validación. Con esta separación se procede a realizar la lectura de los datos simplificados, para ello se carga el archivo de excel que contiene todos los ejemplos con sus respectivas características y en la variable *features* se almacena todo el conjunto; mientras que en *Labels* se almacenan cada una de las etiquetas que asocian a cada ejemplo con la clase correspondiente.

Consecuente a esto se introducen los datos a la función *FeatSelection(features, Labels, NumberComponents)* en la cual entrega los datos de entrenamiento y prueba para ser empleados en el proceso de clasificación. Luego se crea el objeto de la clase svm y se determina que es un modelo lineal con la siguiente función *clf = svm.SVC(kernel='linear')* la cual inicializa el clasificador, con ello se procede a la parte de entrenamiento del mismo, donde se introducen los datos *train* como parámetros de entrada a través del siguiente método $clf.fit(X_{train}, y_{train})$ de la clase SVC.

Finalmente, al entrenar el modelo se procede a realizar una prediccón de los datos, es decir, se determina el porcentaje de probabilidad con que una imagen pertenece a una clase en específica y para esto se realiza la implementación de la siguiente función $y_{pred} = clf.predict(X_{test})$. De igual forma a la regresión logística se calcula la matriz de confusión para poder determinar el desempeño y aprendizaje del algoritmo.

Procesos Gaussianos

Por último, se implementa el clasificador de procesos gaussianos GP el cual otorga el paquete *Scikit Learn*, este modelo de clasificación estaba basado en la aproximación de Laplace. GP es un clasificador binario, sin embargo este paquete permite el uso de métodos como *One vs All* y *One vs One* para la clasificación multiclase. Importando de sklearn el conjunto de métodos para crear un clasificador GP por medio de *from sklearn.gaussian_process import GaussianProcessClassifier*. De igual forma que en los clasificadores anteriores utilizando la matriz de características reducidas, se trabaja con los datos de entrenamiento X_{train} (70 %) y de validación X_{test} (30 %).

Luego se crea el objeto de inicialización del clasificador con *gpc = GaussianProcessClassifier(kernel=kernel,random_state=0,multi_class = 'one_vs_rest')* con varios parámetros de configuración entre los cuales están:

- **kernel:** Se trabaja con el kernel RBF (*Radial Basis Function*) que permite una mejor clasificación de los datos que son no linealmente seperables, este se obtiene a través del paquete *from sklearn.gaussian_process.kernels import RBF* y para la definición del mismo por medio *kernel = 1.0 * RBF(0.9)*. A este se le ingresa como parámetro un valor de inicialización, que con la realización de unas pruebas el mejor resultado se obtuvo con un valor de 0.9.
- **random_state:** Este se establece con el valor de 0, debido a que ya se realiza un proceso aleatorio en los datos en la selección de características.
- **multi_class:** Este permite especificar que se va a tratar de un problema de múltiples clases. Aquí a diferencia de los dos anteriores clasificadores, en lugar de trabajar con el método *One vs All* se implementa el *One vs One* donde se observa unos mejores resultados.

  Debido a las pruebas realizadas comparando las dos métodos multiclase, se obtuvo un mejor resultado en *One vs One* por la distribución de los datos o el número de ejemplos de cada clase para obtener la frontera de clasificación. Como en *One vs All* se entrena k clasificadores binarios a igual número de clases, para cada clasificador binario toma una clase su etiquetas se vuelven $y = +1$ y el de las demás clase se convierten en la etiqueta de la clase contraria $y = -1$. Esto genera una descompensación de la clase haciendo que el clasificador se sesgue, de manera que la tendencia vaya a la clase con la mayor cantidad de ejemplos, realizando clasificaciones erróneas. En cambio en *One vs One* se determina un par de clases de un conjunto de n clases y se determina un clasificador binario para cada par de clases, esto mejora un poco la distribución entre clases obteniendo un mejor resultado.

De acuerdo con lo anterior, el número de clasificadores en *One vs One* se determina a través de la ecuación 2.3 basados en la teoría combinatoria, en la cual con $n = 4$ número de clases, y para la clasificación de un par de clases $r = 2$, se obtiene un total de 6 clasificadores:

$$\binom{n}{r} = \frac{n!}{r!\,(n-r)!} \tag{2.3}$$

Después se ajusta y se entrena el modelo con los datos y etiquetas de entrenamiento *gpc=gpc.fit(Xtrain,Ytrain)*. Finalmente con los parámetros y el modelo entrenado, se realiza las predicciones para los datos de validación *y_pred = gpc.predict(Xtest)* y se construye la matriz de confusión *Conf_Matrix = confusion_matrix(Ytest,y_pred)*. Los resultados son mostrados en el siguiente capítulo.

Capítulo 3

Resultados

3.1. Resultados

En este capítulo se presentan los resultados de cada una de las etapas de preprocesamiento, extracción, selección de características y clasificación, así como el análisis de los resultados de clasificación con la matriz de confusión para cada clasificador y contrastados con los resultados del estado del arte.

Como primer elemento, el resultado de aplicar todo el procedimiento de preparación de la imagen se visualiza en la figura 3.1, este procedimiento consiste en preparar y estandarizar la imagen para su posterior procesamiento. En la figura 3.1 a) se lee la imagen original y b) se pasa a escala de grises para la detección facial, en c) se logra visualizar la ventada de detección facial junto con la detección de cada uno de los ojos. Después se realiza la estandarización al hacer la alineación del rostro en d) y finalmente en e) se tiene el resultado final de esta etapa al recortar solo la sección del rostro y escalar la imagen a 256×256 píxeles.

Esta etapa es de vital importancia para el buen rendimiento de las siguientes etapas de extracción de características y clasificación debido a que muchos algoritmos de reconocimiento y métodos de aprendizaje de máquina se benefician de esta normalización. El objetivo de la alineación, escalado y recorte es el de normalizar y estandarizar la imagen para que las mayores variaciones presentes en el rostro sean características relacionadas con la edad, al tener todos imágenes en la misma orientación [7] [9] [11].

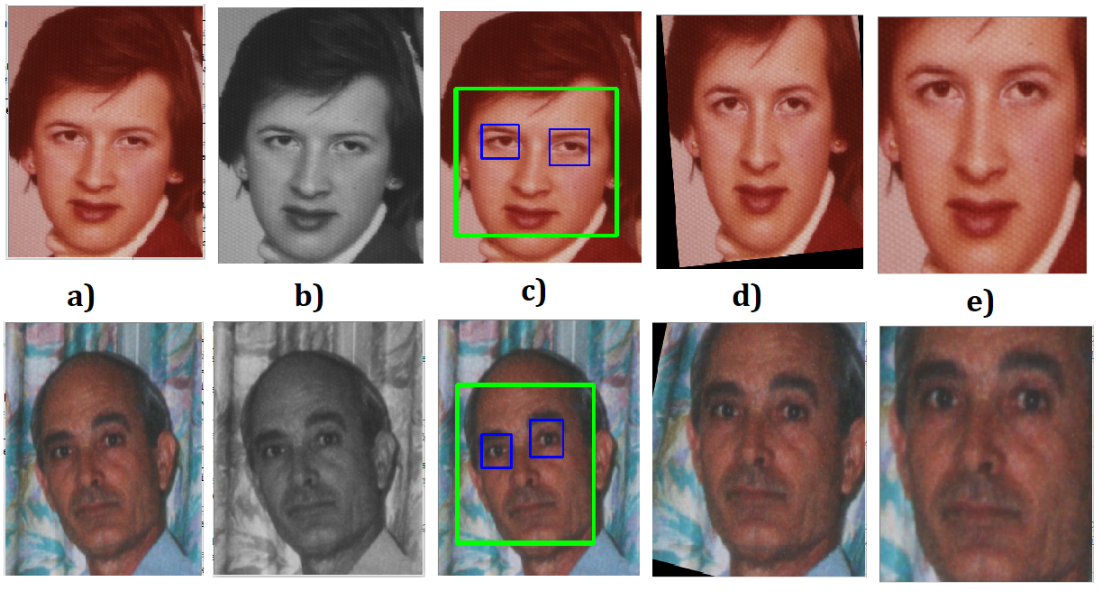
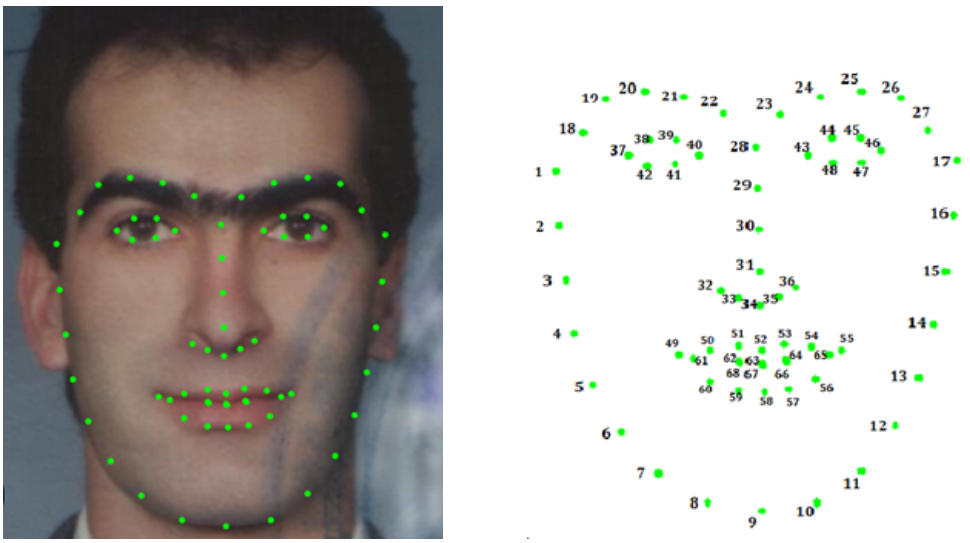


Figura 3.1: Resultados del preprocesamiento, a) Imagen de entrada (original), b) Imagen en escala de grises, c) Detección del rostro y ojos, d) Alineación o rotacón, e) Recorte y escalado.

En la etapa de extracción de características se busca a partir de la imagen preprocesada obtener un vector de datos que represente y codifique las características más relevantes para determinar la edad, como el crecimiento craneofacial, la textura de la piel, las arrugas y manchas. Es así que con el descriptor AAM se detectan las estructuras faciales más importantes para identificar el crecimiento craneofacial tal y como se visualiza en la figura 3.2. Como resultado de este descriptor se obtiene un vector F_{AAM} con 136 características que corresponden a la dupla (x,y) de coordenadas de cada punto de referencia (*facial landmark*).

$$F_{AAM} = \{x_1, y_1, x_2, y_2, ..., x_{68}, y_{68}\}_{1\times136}$$



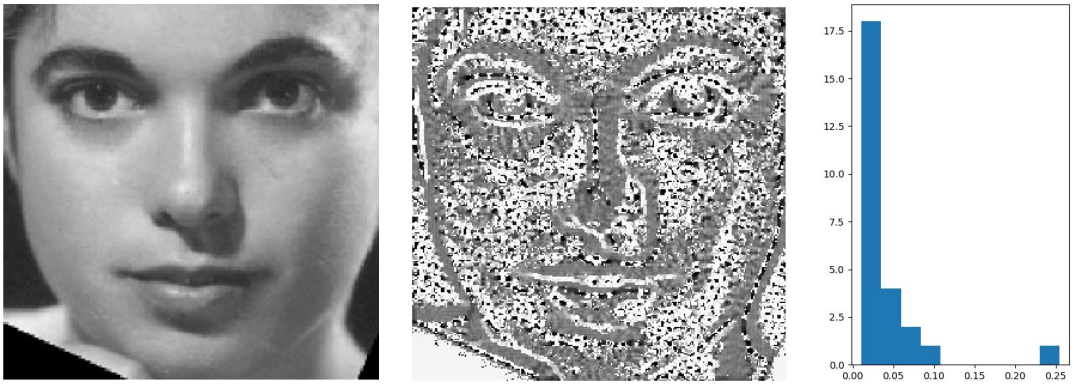
(a) Imagen de ejemplo con la detección de los puntos.

(b) Enumeración de cada uno de los puntos del rostro.

Figura 3.2: Puntos de referencia.

Continuando con el descriptor LBP el cual tiene como objetivo describir texturas y patrones para así codificar contornos y puntos. Con la característica que es robusto a cambios de iluminación, ruido e invariante a la escala de grises y a la rotación. En la figura 3.3 se observa el resultado de aplicar LBP a una imagen facial. LBP se calcula con $p = 24$ puntos y con un radio de $r = 3$, por lo tanto se obtiene un vector F_{LBP} con 26 características.

$$F_{LBP} = \{h_1, h_2, ..., h_{27}\}_{1\times26}$$



(a) Imagen preprocesada en escala de grises.

(b) Característica LBP.

(c) Histograma del patrón.

Figura 3.3: Resultado del descriptor LBP.

La transformada Wavelet permite el realce de bordes, el análisis de textura, y la extracción de detalles al aplicarle a la descomposición de la imagen unos filtros (lh, hl, hh) obteniendo detalles horizontales, verticales y diagonales, lo que permite denotar ciertos patrones como las arrugas. Para este resultado se obtiene un vector F_{Wav} con 12 características, que corresponden a los 4 momentos estadísticos (media, desviación, sesgo y kurtosis) aplicado a los 3 respuestas de los filtros.

$$F_{Wav} = \{mlh, stdlh, ..., shh, khh\}_{1\times 12}$$

El último descriptor *Gabor-wavelet* detecta bordes y posee propiedades de localización tanto en el dominio espacial como en el dominio de la frecuencia. Al variar las escalas y orientaciones de los filtros tiene como objetivo resaltar características como diferentes arrugas de distintos tamaños y orientaciones [26] [27]. Tal y como se observa en la figura 2.4 de la sección de metodología a las 40 respuestas se obtienen la media y la varianza para codificar la información relacionadas con las intensidades y orientaciones de las arrugas [9] [16]. Formando el vector F_{Gabor} con 80 características en total.

$$F_{Gabor} = \{w_1, w_2, ..., w_{80}\}_{1\times 80}$$

Ahora se concatena los vectores de características, para formar el vector híbrido con las características globales y locales, obteniendo un robusto descriptor de forma y textura. Es así que se obtiene el vector final $F = \{F_{AAM}, F_{LBP}, F_{Wav}, F_{Gabor}\}_{1\times 254}$ con 254 características en total. Ahora obteniendo el vector de características para cada una de las imágenes de la base de datos se construye la matriz de características X de dimensiones 1345×254 ($N \times D$), en donde las filas corresponden a los ejemplos o imágenes y las columnas son el número de características.

3.1.1. Selección de características

Realizando la selección de características más relevantes y con mayor variabilidad con el objetivo de reducir el ruido y hacer que otros algoritmos de clasificación funcionen mejor al reducir la cantidad de características, mejorando la eficiencia computacional y evitando el *overfitting* (sobre-entrenamiento) de los datos con los que se entrena el modelo de clasificación, haciéndolo más general [31] [32]. En la figura 3.4 se observa la gráfica de rendimiento vs número de componentes (PC) como se mencionó en la metodología. Con este procedimiento se permite apreciar en la gráfica que hay un crecimiento sustancial en el rendimiento hasta $PC = 50$, a partir de este punto, con un mayor número de características las variaciones de rendimiento son mínimas.

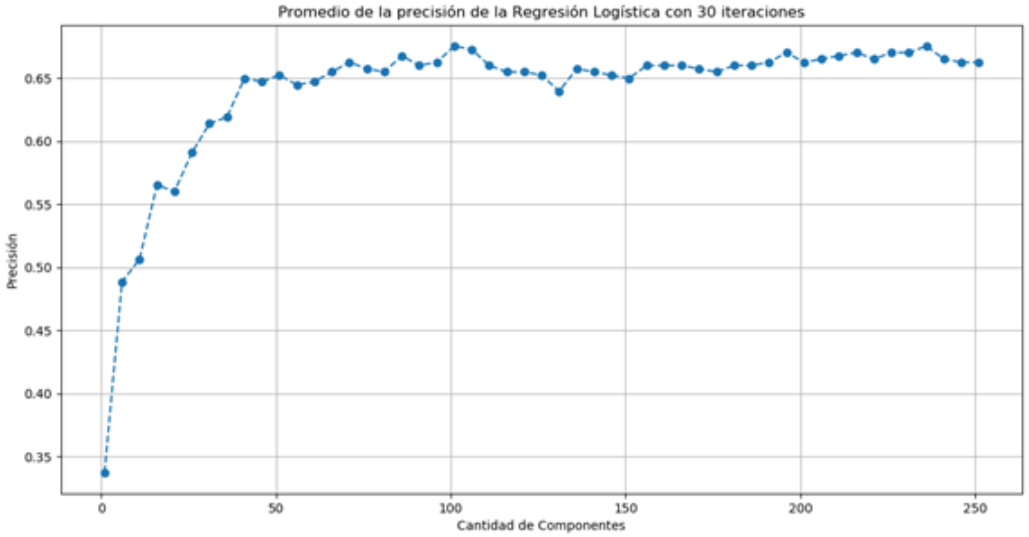


Figura 3.4: Promedio del rendimiento del clasificador de prueba variando el número de componentes PC.

Cabe aclarar que la gráfica de la figura 3.4 es construida con la matriz de características en donde el descriptor AAM se aplica a la imagen original de la base de datos sin alineación. Por esta razón para mejorar el rendimiento y apreciar la importancia de la alineación y escalado de la imagen, en la figura 3.5 la gráfica se construyó basándose en la matriz de características con AAM con alineación de pose y los demás descriptores con las imágenes preprocesadas. Se logra observar una mejoría en la precisión y rendimiento de clasificación para cada componente PC. Logrando llegar a un rendimiento del 70 %

Figura 3.5: Promedio del rendimiento del clasificador de prueba variando el número de componentes PC con alineación de pose para el descriptor AAM.

Por lo tanto a partir de la gráfica de la figura 3.5 se escoge como número de componentes optimas $PC = 95$ para realizar la reducción a la matriz de características. La nueva matriz de características (X) para entrenar y validar los modelos de clasificación es de dimensiones 1345×95 $(N \times K)$.

3.1.2. Clasificación

En esta etapa de clasificación se divide la matriz de características en dos conjuntos de datos, 70 % de los ejemplos para el conjunto de entrenamiento del modelo y 30 % para realizar la validación. Esto con el objetivo de ver la generalización que tienen los clasificadores al probarse con un conjunto de datos diferentes al usado para el entrenamiento. Para las pruebas de validación se construye la matriz de confusión que permite realizar un análisis completo del desempeño y fiabilidad de los clasificadores, a partir de las predicciones de etiqueta que arroja el clasificador y de las etiquetas verdaderas (obtenidas de la edad verdadera de la persona en la imagen). En esta matriz de confusión las columnas corresponden a las etiquetas reales de cada clase y las filas son los resultados de clasificación (etiquetas entregadas por el clasificador).

En la tabla 3.1, 3.2 y 3.3 se observan los resultados de la matriz de confusión obtenidos de la prueba con el conjunto de validación para cada uno de los métodos de clasificación, regresión logística, SVM y procesos gaussianos (GP) respectivamente. En la diagonal de la

matriz de confusión se presentan las imágenes que fueron correctamente clasificadas en sus clases. Cabe aclarar que estos resultados se obtuvieron tras realizar 30 iteraciones del proceso de clasificación y determinar el promedio de la matriz de confusión.

		Clase verdadera			
		1	**2**	**3**	**4**
Predicción	**1**	87.8 %	7.65 %	3.73 %	0.81 %
	2	27.5 %	45.58 %	23.52 %	3.39 %
	3	10.05 %	21.98 %	38.56 %	29.42 %
	4	1.38 %	2.43 %	16.6 %	79.58 %

Cuadro 3.1: Matriz de confusión con Regresión Logística.

		Clase verdadera			
		1	**2**	**3**	**4**
Predicción	**1**	86.66 %	9.02 %	3.78 %	0.53 %
	2	28.31 %	45.95 %	22.31 %	3.43 %
	3	12.95 %	22.56 %	35.35 %	29.13 %
	4	1.84 %	3.77 %	17.09 %	77.3 %

Cuadro 3.2: Matriz de confusión con SVM.

		Clase verdadera			
		1	**2**	**3**	**4**
Predicción	**1**	87.10 %	7.51 %	4.6 %	0.79 %
	2	29.9 %	44.55 %	23.58 %	5.97 %
	3	10.74 %	20.64 %	36.87 %	31.74 %
	4	1.66 %	3.83 %	16.04 %	78.47 %

Cuadro 3.3: Matriz de confusión con GP.

Primero para el clasificador con regresión logística se tiene una exactitud del 68,5 % al clasificar 267 imágenes en la clase correcta de 390 imágenes de prueba. Analizando los resultados de cada una de las clases, en la clase de niños (clase 1) se tiene un muy buen rendimiento al clasificar 125 imágenes correctas de 142 (rendimiento del 87,8 %), al igual

que en la clase de adultos mayores (clase 4) que clasifico correctamente 81 imágenes de 101 (rendimiento del 80 %). Caso contrario con las dos clases internas, en la cual la clase de jóvenes (clase 2) presenta 32 imágenes correctas de 69 dando un rendimiento del 45,5 %, y en la clase de adultos (clase 3) se clasificaron correctamente 29 imágenes de 76 con un rendimiento del 38,5 %.

De igual forma, en la tabla 3.2 para SVM se tiene una exactitud global del 66,8 %, al clasificar correctamente 260 imágenes en la clase correspondiente de 390. Se conserva el mismo comportamiento al clasificar correctamente las clases 1 y 4 con rendimientos mayores de 86,66 % y 77,3 % respectivamente. Mientras que en las clases intermedias (clases 2 y 3) se tiene un rendimiento del 45,9 % y 35,35 % respectivamente. Finalmente en la tabla 3.3 se observan los resultados obtenidos con GP el cual presenta un rendimiento general del 66,48 % con kernel RBF y con el método *One vs One*; clasificando 259 imágenes en las clases correctas de 390 y el rendimiento de cada clase se visualiza en la tabla.

El error de clasificación presentado en las clases 2 y 3 (las de menor rendimiento) se confunden entre las clases vecinas 1, 2 y 3. Esto se debe a que el rostro de la persona en estas edades tiene pocas arrugas, sin manchas en la piel debido a una buena genética, estilo de vida, salud, maquillaje o cirugías plásticas, etc. Además, la confusión también se debe a los límites entre las fronteras con ejemplos difíciles de clasificar [9] tal y como se visualiza en la figura 3.6. Por ejemplo, para una imagen de una persona con edad entre los 22 y 23 años (pertenece a la clase 3) fácilmente puede ser clasificada como de clase 2 (entre los 14 - 21 años).

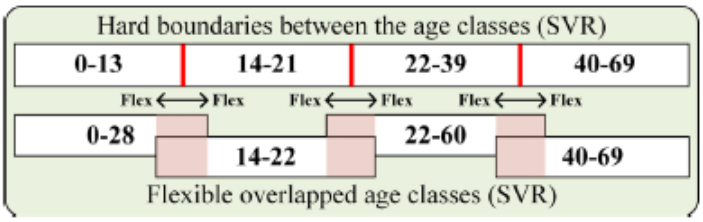


Figura 3.6: Clasificación jerárquica de la edad usando fronteras flexibles en la etapa de regresión (tomado de: [9]).

En el estado del arte las clasificaciones erróneas en los límites del grupo se compensan en la etapa de regresión superponiendo los rangos de edad en las clases vecinas [35]. Es decir, relajan o vuelven flexible la frontera en el segundo nivel de clasificación y consideran rangos de edad superpuestos en la etapa de regresión cuando estiman el valor de la edad, para reducir la clasificación errónea (las clases superpuestas se definen para cada grupo de edad y abarcan un rango mayor para su posterior estimación) [9].

En contraste con los resultados del estado del arte en [9] con un clasificador SVM, entrenado con diferentes combinaciones de características locales y globales. Trabajando con un kernel

RBF (*Radial Basis Function*) tuvo una exactitud de 70 % mientras que con un kernel lineal la mayor exactitud corresponde a 68, 8 %. En cambio en [35] con un clasificador de procesos gaussianos (GP) con características obtenidas con HOG, muestra una exactitud general de 56,07 %, pero presenta un mejor rendimiento en comparación con [9] en la última clase (clase 4 de 22 - 69 años) de 44,92 % tal y como se observa en la tabla 3.4.

La diferencia de tasa de acierto y error en los modelos de la tabla 3.4 ([9] y [35]), además de las diferentes técnicas de extracción de características y clasificación, también se deben a los rangos de edad y a la cantidad de imágenes para entrenamiento y test de cada clase. En donde en [9] utilizaron solo FGNET y la clase 1 quedo con más del 50 % de las imágenes por tal razón obtuvo el mayor rendimiento en esa clase, y en las otras clases se obtuvo un error de clasificación alto, al tener pocas imágenes para entrenamiento y test. Mientras que en [35] para FGNET dividieron los rangos de edad de tal forma que la distribución de las edades para cada clase fue equilibrada obteniendo un mejor rendimiento en las clases 2, 3 y 4, aun así en [35] la clase 4 al abarcar un gran rango de edad (22 - 69 años), el rendimiento fue menor comparado con el método propuesto en este documento, el cual presenta una muy buena exactitud del 80 % para la clase 4, debido a que el rango de edad es menor y que se compensa para esta clase con más imágenes de MORPH-II, logrando una distribución más equilibrada.

Rendimientos	General	Clase 1	Clase 2	Clase 3	Clase 4
SVM+SVR [9]	70 %	92.2 %	29.3 %	29.8 %	5.9 %
Jerárquico GP [35]	56.07 %	56.65 %	74.59 %	49.06 %	49.92 %
Propuesto Regresión Logística	**68 %**	**89.1 %**	**52 %**	**33 %**	**80 %**
Propuesto SVM	66.8 %	86.66 %	45.95 %	35.35 %	77.3 %
Propuesto GP	66.48 %	87.10 %	44.55 %	36.87 %	78.47 %

Cuadro 3.4: Tasa de acierto y error en los modelos .

Capítulo 4

Conclusiones

En este trabajo se presenta un algoritmo para el reconocimiento de grupos etarios con un enfoque basado en características de forma (globales) y en textura (locales) para la representación del envejecimiento del rostro humano. Durante el desarrollo se logró evidenciar aspectos relevantes para tener un buen rendimiento en métodos de aprendizaje de máquina y específicamente en este problema, entre estos están:

Los efectos que son corregidos en la etapa de preprocesamiento como las variaciones de pose, escala y traslación, permiten un mejor rendimiento de las siguientes etapas del proceso de clasificación de edad, tal y como se aprecia en la figura 3.5 el rendimiento de clasificación mejora respecto al de la figura 3.4 con aplicarle la alineación del rostro al descriptor faltante (AAM). Lo anterior permite evidenciar y concluir el mejoramiento de las tareas de clasificación con una buena práctica de normalización previa como la alineación y el escalado, dejando solo las características más importantes relacionadas con la edad.

En la etapa de selección de características una reducción de dimensionalidad puede afectar positiva o negativamente los resultados de clasificación, por lo cual con la metodología implementada en este trabajo se logra a partir de la figura 3.5 visualizar y escoger el óptimo número de características que represente la mayor variabilidad de los datos originales teniendo en cuenta la relación entre exactitud y costo computacional.

En los resultados de clasificación se logra observar una buena discriminación entre las clases de niño y adulto mayor, lo que indica la efectividad del descriptor de forma (características globales) para separar la clase niño de las demás, al modelar el crecimiento craneofacial siendo más notorio en la etapa de la niñez. De igual forma los descriptores de textura (características locales) son efectivos al modelar las arrugas y texturas de la piel logrando discriminar bien la clase de los adultos mayores de las otras. Sin embargo debido a los bajos rendimientos de las clases intermedias (joven y adulto) se concluye que las características de forma y textura

no son lo suficientemente discriminatorias debido a que en estos rangos de edad la forma y la textura de la piel son similares, además de factores como el maquillaje o cirugías estéticas impactan negativamente la estimación de la edad.

Capítulo 5

Trabajo futuro

A partir de este trabajo es posible implementar algunos aspectos que se pueden implementar en el futuro:

- Implementar un algoritmo de estimación de edad usando la información de género proporcionada por otro clasificador, es decir diseñar un clasificador de género y luego tener un clasificador de edad para cada género, esto se debe a que tanto el hombre como la mujer tienen procesos de envejecimiento diferentes, por lo tanto se puede ver beneficiado el rendimiento respecto con la información de género.
- Implementar un algoritmo de clasificación jerárquico de dos niveles, uno de clasificación en grupos etarios y el segundo nivel de regresión basados en cada uno de los grupos de edad, para la estimación del valor de la edad. Se puede lograr resultados positivos y una mejora en el rendimiento al superponer las fronteras en la etapa de regresión y al trabajar con un enfoque jerárquico.
- Además de los descriptores de forma y textura mencionados en este trabajo, implementar otro tipo de descriptor para mejorar el rendimiento de las clases intermedias (jóvenes y adultos) en la cual se modele otro tipo de característica que permita discriminar mejor estas dos clases. De igual forma trabajar con clasificadores con flexibilidad en la frontera para mejorar el rendimiento para los casos que se encuentre entre los límites de decisión de clases vecinas.

Bibliografía

[1] C. O. B. K. A. J. L. Best-Rowden, H. Han, "Unconstrained face recognition: identifying a person of interest from a media collection," *IEEE Trans. Inf. Forensics Secur*, pp. 2144–2157, 2014.

[2] C. D. G. Guo, Y. Fu and T. Huang, "Image-based human age estimation by manifold learning and locally adjusted robust regression," *IEEE Trans. Image Processing*, vol. 17, no. 7, pp. 1178–1188, Jul. 2008.

[3] A. J. H. Han, C. Otto, "Age estimation from face images: human vs. machine performance, in: Proceedings of the international conference on biomet- rics (icb)," *IEEE*, pp. 1–8, 2013.

[4] G. G. Y. Fu and T. S. Huang, "Age synthesis and estimation via faces: A survey," *IEEE Trans. Pattern Anal. Mach. Intell*, vol. 32, no. 11, pp. 1955–1976, Nov. 2010.

[5] B. F. H. Chen, Y. Yan Tang and T. Zhang, "Wavelet-based multi-level image matching with detail measure weight for face recognition under varying illumination," *IEEE College of Comp. Science, Chongqing University, Chongqing 400044*, 2010.

[6] M. S. B. S. Zaghbania, N. Boujnehb, "Age estimation using deep learning," *ScienceDirect Computers and Electrical Engineering*, no. 68, pp. 337–347, Abr. 2018.

[7] J. F. d. P. J. M. C. A. González Briones, G. Villarubia, "A multi-agent system for the classification of gender and age from images," *ScienceDirect Computer Vision and Image Understanding*, 2018.

[8] R. E. E. Eidinger and T. Hassner, "Age and gender estimation of unfiltered faces," *IEEE Trans. On Inf. Forensics and Sec.*, vol. 9, no. 12, pp. 2170–2179, Dic. 2014.

[9] C. A. L. J. K. Pontesa, A. S.Britto Jr, "A flexible hierarchical approach for facial age estimation based on multiple features," *Pattern Recognition*, vol. 54, pp. 34–51, Jun. 2016.

[10] S. D. P., "Reconocimiento facial mediante el análisis de componentes principales pca," *Escuela Técnica Superior de Ingeniería, Universidad de Sevilla*, 2017.

[11] C. G. Turhan and H. S. Bilge, "Class-wise two-dimenasional pca method for fase recognition," *IET Comput. Vis.*, vol. 11 Iss 4, pp. 286–300, Sep. 2016.

[12] Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features. in computer vision and pattern recognition," *IEEE Computer Society Conference*, 2001.

[13] P. I. Wilson and J. Fernández, "Facial feature detection using haar classifiers," *Texas University Corpus Christi*, Abr. 2006.

[14] Y. Freund and R. Schapire, "A short introduction to boosting," *Journal of Japanese Society for Artificial Intelligence*, 1999.

[15] Z. Hu and Y. Wen, "Facial age estimation with age difference," *IEEE Transactions on Image Processing*, vol. 26, pp. 3087–3097, Dic. 2016.

[16] Y. . L. S. . P. K. . K. J. Choi, Sung & Lee, "Age estimation using a hierarchical classifier based on global and local facial features," *Pattern Recognition*, no. 44, pp. 1262–1291, 2011.

[17] N. S. Lakshmiprabha, "Face image analysis using aam, gabor, lbp and wd features for gender, age, expression and ethnicity classification," *ArXiz*, 2016.

[18] Z.-H. Z. X. Geng and K. Smith-Miles, "Automatic age estimation based on facial aging patterns," *IEEE Trans. Pattern Anal. Mach. Intell*, vol. 29, no. 12, pp. 2234–2240, Dic. 2007.

[19] D. . Y. J. Wang, Shengzheng & Tao, "Relative attribute svm+ learning for age estimation," *IEEE transactions on cybernetics*, no. 46, 2015.

[20] T. D. B. K. Luu, K. Ricanek and C. Y. Suen, "Age estimation using active appearance models and support vector machine regression," *IEEE 3rd International Conference on Biometrics: Theory, Applications, and Systems*, pp. 1–5, 2009.

[21] G. . T. C. Cootes, T.F. & Edwards, "Active appearance models. pattern analysis and machine intelligence," *IEEE Trans.*, no. 23, pp. 681–685, 2011.

[22] M. P. Timo Ojala and T. Mäenpää. Multiresolution gray scale and rotation invariant texture classification with local binary patterns. [Online]. Available: http://www.outex.oulu.fi/publications/pami_02_opm.pdf

[23] H. F. Zhibin Pan and L. Zhang, "Texture classification using local pattern based on vector quantization," *IEEE Trans. in Image Processing*, vol. 24, no. 12, pp. 5379–5388, Dic. 2015.

[24] N. M. A. M. A. M. S. M. A. M. M. G. M. Safrrin Karis, N. Rafiqah Abdul Razif, "Local binary pattern (lbp) with application to variant object detection: A survey and method," *Mohd Safrrin Karis, Nur Rafiqah Abdul Razif, Nursabillilah Mohd Ali, M. AsyrafRosli, Mohd Shahrieel Mohd Aras, Mariam Md Ghazaly*, pp. 221–226, Mar. 2016.

[25] Y. MA, "Number local binary pattern: an extended local binary pattern," *Department of Information Engineering, Lanzhou University of Finance and Economics*, pp. 272–275, Jul. 2011.

[26] C. Liu and H. Wechsler, "Gabor feature based classification using the enhanced fisher linear discriminant model for face recognition," *IEEE Transactions on Image Processing*, vol. 11, no. 4, pp. 467–476, Abr. 2002.

[27] H. S. Saeed Meshgini, Ali Aghagolzadeh, "Face recognition using gabor-based direct linear discriminant analysis and support vector machine," *Computers & Electrical Engineering*, vol. 39, no. 3, pp. 727–745, 2013.

[28] P. B. Dr. Sudeep D. Thepade, "Iris recognition using fractional coefficients of transforms, wavelet transforms and hybrid wavelet transforms." *International Conference on Control, Computing, Communication and Materials*, 2013.

[29] I. Bayram and I. W. Selesnick, "Frequency-domain design of overcomplete rational-dilation wavelet transforms," *IEEE Trans. on signal processing*, vol. 57, no. 8, pp. 2957–2972, Ago. 2009.

[30] Sebastian. Principal component analysis in python/v3. [Online]. Available: https://plot.ly/python/v3/ipython-notebooks/principal-component-analysis/?fbclid=IwAR2MsekF4B43s29wXzVpdyikIgZ1h-K9oLO1N_XWF4eOh7YjybHTaJkceJs

[31] M. Cavaioni. Machine learning: Unsupervised learning - principal component analysis. [Online]. Available: https://medium.com/machine-learning-bites/machine-learning-unsupervised-learning-principal-component-analysis-8f7ad311027e

[32] R. M. Ebied, "Feature extraction using pca and kernel-pca for face recognition," *IET Computer Vision*, pp. 286–300, Sep. 2016.

[33] J. C. Bedoya, "Algoritmos para localización de fallas en sistemas de distribución usando máquinas de soporte vectorial," *Maestría en ingeniería eléctrica. Universidad Tecnológica de Pereira*, pp. 15–20, 2010.

[34] M. V. A. López, E Valveny, "Curso de detección de objetos," *Universidad Autónoma de Barcelona*. [Online]. Available: https://www.coursera.org/learn/deteccion-objetos/home/info

[35] M. M. Sawant and K. Bhurchandi, "Hierarchical facial age estimation using gaussian process regression," *in IEEE Access*, vol. 7, pp. 9142–9152, 2019.

[36] O. Knagg, "An intuitive guide to gaussian processes." [Online]. Available: https://towardsdatascience.com/an-intuitive-guide-to-gaussian-processes-ec2f0b45c71d

[37] C. E. Rasmussen and C. K. I. Williams, "Gaussian processes for machine learning," *Cambridge, MA, USA: MIT Press*, vol. 1, 2016.

[38] Y. Zhang and D. Yeung, "Multi-task warped gaussian process for personalized age estimation," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 7, pp. 2622–2629, 2010.

[39] D. L. Zan Yang, "Application of logistic regression with filter in data classification," *IEEE - Department of Science, Tongji Zhejiang College*, pp. 3755–3759, Jul. 2019.

[40] R. E. S. MICHAEL COLLINS, "Logistic regression, adaboost and bregman distances," *Machine Learning - AT&T Labs - Research, Shannon Laboratory*, pp. 253–285, 2002.

[41] A. links open overlay panelJianboYu, "State of health prediction of lithium-ion batteries: Multiscale logic regression and gaussian process regression ensemble," *Reliability Engineering & System Safety*, vol. 174, pp. 82–95, Jun.

[42] Q. W. Hui Zhang, Ping Chen, "Fault diagnosis method based on eemd and multi-class logistic regression," *International Conference on Smart City and Systems Engineering (ICSCSE)*, pp. 859–863, 2018.

[43] J. L. H. J. Jonghee Kim, Jonghwan Lee, "Optimal feature selection for pedestrian detection based on logistic regression analysis," *International Conference on Systems, Man, and Cybernetics*, pp. 239–242, 2013.

[44] Y. Fu. The fg-net aging database. [Online]. Available: http://yanweifu.github.io/FG_NET_data/index.html

[45] U. of North Carolina Wilmington. Morph facial recognition database. [Online]. Available: https://uncw.edu/oic/tech/morph.html

[46] J. Hua. Age estimation based on support vector regression. [Online]. Available: https://github.com/huajh/Age_Estimation_via_fastAAMs

[47] P. oficial de Python. [Online]. Available: https://www.python.org/

[48] P. oficial de OpenCV. Open source computer vision library. [Online]. Available: https://opencv.org/about/

[49] A. Rosebrock. Face alignment with opencv and python. [Online]. Available: https://www.pyimagesearch.com/2017/05/22/face-alignment-with-opencv-and-python/

[50] M. F. LinLin Shen, Li Bai, "Gabor wavelets and general discriminant analysis for face identification and verification," *Image and Vision Computing*, vol. 25, no. 5, pp. 553–563, 2007.